M 2014

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

NEUROGAIT

MOBILE RECORDING OF GAIT IN PATIENTS WITH NEUROLOGICAL DISEASES

PEDRO MANUEL VAZ DUARTE OLIVEIRA E SÁ

DISSERTAÇÃO DE MESTRADO APRESENTADA
À FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO EM
ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES 2013/2014

A Dissertação intitulada

"NeuroGait: Mobile Recording of Gait in Patients with Neurological Diseases"

foi aprovada em provas realizadas em 09-10-2014

o júri

Presidente Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Joel José Puga Coelho Rodrigues
Professor Auxiliar do Departamento de Informática da Universidade da Beira Interior

Professor Doutor João Paulo Trigueiros da Silva Cunha
Professor Associado do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Pedro Manuel Vaz Duarte Oliveira e Sá

Autor - Pedro Manuel Vaz Duarte Oliveira e Sá

Faculdade de Engenharia da Universidade do Porto

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

NeuroGait: Mobile Recording of Gait in Patients with Neurological Diseases

Pedro Sá

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: João Paulo da Silva Cunha (Prof. Dr.)

Co-supervisor: Kai Bötzel (Prof. Dr.)

October 29, 2014

Resumo

Esta dissertação foca-se no desenvolvimento de um sistema de quantificação de marcha sem fios que utiliza sensores inerciais (IMUs) denominados de Motion Motes (ou MoMos) emparelhados com um dispositivo Android.

Este sistema recolhe dados sincronamente de até quatro MoMos e tem também a capacidade de gravar vídeo em alta-definição simultaneamente.

O objetivo principal deste trabalho foi desenvolver uma plataforma que pudesse auxiliar ao diagnóstico da Doença de Parkinson e avaliação do estado dos pacientes que fosse barato, fácil de usar, não-obtrusivo e que pudesse ser usado em contexto ambulatório.

Apesar de ter sido desenvolvido para a Doença de Parkinson, a plataforma também é potencialmente útil em outras aplicações.

De forma a validar este sistema, quatro MoMos foram colocados em cada coxa e canela de 10 voluntários saudáveis a quem foi pedido para andar num tapete rolante a velocidades diferentes. Um sistema de captura de movimento Oqus da Qualisys foi usado como referência.

Testes meticulosos à perfomance do sistema, como a medição de atrasos entre o início da gravação dos MoMos e do vídeo foram também realizados e analisados em detalhe.

As características deste sistema - a capacidade de reunir dados sincronamente de quatro IMUs num disposito Android e simultaneamente gravar vídeo - tornam-no num sistema único dentro dos instrumentos de avaliação de marcha existentes.

Abstract

This dissertation focuses on the development of a wireless gait quantification system using Bluetooth-enabled inertial measurement units (IMUs) called Motion Motes, or MoMos, connected to an Android phone.

This system synchronously gathers data from up to four MoMos and also has the ability to record HD video simultaneously. The goal was to design a platform to assist clinicians on Parkinson's Disease patients diagnosis and state assessment that is inexpensive, easy to use, unobtrusive and can inherently be used in an ambulatory setting.

Although designed with PD evaluation in mind, the platform is potentially useful for other applications as well.

To validate this system, 4 MoMos were placed on each thigh and shank of 10 healthy male subjects who were asked to walked on a treadmill at different speeds. A Qualisys Oqus camera motion capture system was used as ground truth.

Extensive tests on the system's performance, such as the evaluation of recording delays between IMUs and video were also performed and thoroughly analyzed.

This system's features - the capability of synchronously gathering data from four IMUs onto Android device and simultaneously record video - make it an unique system within the gait assessment instruments already in existence.

Acknowledgements

I'd like to express my deep gratitude towards my family, who provided me with an unsurmountable amount of support throughout my degree and granted me the gift of education.

To my girlfriend, who provided me with love, care and expertise throughout this thesis.

I'd also like to thank my supervisor Prof. Dr. João Paulo da Silva Cunha for his guidance, availability and for providing all the means I could ever need to successfully complete this work.

I also wish to thank my co-supervisor Prof. Dr. Kai Bötzel who allowed me to observe first hand the effects of Parkinson's Disease and introduced me to who this work is intended to help, which served as an amazing motivator; and for providing all the means and resources to evaluate this system.

To my friends at BRAIN Lab who worked besides me everyday and offered precious inputs, thank you.

To my friends and colleagues who accompanied me in this journey, thank you all. And to Ricardo Ferreira, who thoroughly proof read this thesis, thank you.

To all, my sincerest and deepest gratitude!

Pedro Sá

"The cosmos is within us.
We are made of star-stuff.
We are a way for the universe to know itself."

Carl Sagan

Contents

List of Figures

List of Tables

Abbreviations

Abreviaturas e Símbolos

ADT	Android Developer Tools
Acc	Accelerometer
App	Application
FoG	Freezing of Gait
g	Acceleration of the Earth's gravity ($9,81\ ms^{-2}$)
Gyr	Gyroscope
IMU	Inertial Measurement Unit
MAC	Medium Access Control
Mag	Magnetometer
Mocap	Motion capture
MoMo	Motion Mote
PD	Parkinson's Disease
STN-DBS	Subthalamic nucleus deep brain stimulation
UI	User Interface

Chapter 1

Introduction

In this chapter, an introduction to the work performed throughout this dissertation is presented. Its content is briefly presented and its structure is detailed.

1.1 Motivation

Parkinson's Disease is one of the most prevalent pathologies in the elderly population. It causes several motor disturbances that can lead to a severely impaired quality of life and increased morbidity.

The lack of quantitative diagnosis and evaluation tools poses a problem in the assessment of a patient's condition, which is fundamental in determining a correct course of treatment. Current methods are largely based on the clinician's analysis and suffer from inter-rater reliability issues.

One of the most important factors in the assessment of PD's is the patient's gait. Current quantitative methods for gait assessment are often costly and can only be operated under laboratorial conditions.

Over the past decade, gait analysis systems based on inertial sensors have arisen and are being shown to be reliable gait assessment tools, which have the potential to aid clinicians perform better decisions regarding patient treatment.

The development of low-cost and unobtrusive solutions that patients can wear is warranted. The integration of inertial sensors with a mobile device seems to be an ideal approach to long-term ambulatory gait quantification systems.

1.2 Objectives

The main objective of this dissertation is to build a synchronous gait capture system using a set of inertial-measurement units (IMUs) connected to an Android device that serves as means to capture the gait data streamed by the IMUs. The system also features simultaneous high-definition video capture, enabling image review of gait events along with IMU data.

These characteristics make the system unique and potentially helpful to clinicians and patients, as it is capable of easily and inexpensively provide valuable insights into a patient's condition. The synchronous video recording combined with the data from the MoMos have the potential to yield unique quantifiable results extracted from human gait.

The work of this thesis focuses on the development of the mobile application that enables the use of IMUs, with an easy user experience.

The general goals for this dissertation are:

- Review of existing solutions of quantitative gait analysis
- Development of a mobile application that integrates with existing IMUs
- Testing of developed application
- System testing with gait data against a ground truth system
- Analysis of obtained results

1.3 Document Outline

In this thesis, the development of the proposed gait analysis system is presented.

Firstly in Chapter 2, a state of the art review is presented, focusing on describing Parkinson's Disease and the current methods available for PD assessment.

In Chapter 3 an overview of the proposed system and details of its architecture is presented.

Chapter 4 contains an in-depth analysis on the supplied hardware, the IMUs, called MoMos that are utilized to capture human gait data.

In Chapter 5, a thorough description and explanation of the mobile application and its development stages are provided.

In Chapter 6 the results for a wide array of tests that assess the application's performance regarding data synchronicity and delays are presented.

In Chapter 7, an analysis on gait highlighting the swing and support phases using the provided equipment is detailed and its results are presented.

In Chapter 8, an overview of the experiments performed at the University Clinic Großhadern in Munich's Ludwig-Maximilians-University to evaluate the NeuroGait system with various subjects is presented. Some results from those experiments are also shown.

Lastly, in Chapter 9, the conclusions for this dissertation are presented - the detailing of the goals achieved and suggested future work.

Chapter 2

State of the Art

This state of the art review focuses firstly on the portrayal of Parkinson's Disease - its prevalence, symptoms, treatment and other topics, and then on the quantitative methods for PD's assessment.

It focuses primarily on understanding gait and the gait disturbances caused by PD.

2.1 Parkinson's Disease

Parkinson's Disease is a chronic neurodegenerative disease first identified in 1817 by James Parkinson [1].

Its prevalence increases with age, for both men and women. It affects about 5% of adults over the age of 80 [2]. There is no cure available and the causes for PD are unknown, as its believed to be a multifactorial disease which may be caused by both genetic and environmental factors [3].

It presents several motor disturbances such as bradykinesia, which is the slowness of movement, hypokinesia, or reduced movements, tremors, freezing of gait and postural instability [4].

One of the most severe impairments with PD are the disturbances in gait, as they severely impact the mobility of patients, reducing quality of life and increasing morbidity, as the risk of falling episodes greatly increase as the disease progresses [5].

It is also a costly disease, as a study conducted in six countries showed that the costs, pertaining medication, patient care and indirect costs for a period of six months ranged from €2620 in Russia to €9820 in Austria [6].

2.1.1 Diagnosis and Assessment

PD is usually assessed using rating scales that evaluate the symptoms and the degree of impairment caused by the disease. Three rating scales - CURS, NUDS, and UPDRS, were found to be the most reliable [7]. However, these evaluations are often somewhat subjective, suffering from interobserver reliability which was found to range from good to moderate depending on the scales used [7, 8].

Accurate evaluations are critical for patient outcomes, as the course of treatment is defined by these evaluations. Therefore, there is a necessity for assessment methods that provide more

objective and reliable inferences on a patient's condition [9, 10]. The main goal of this thesis is to contribute to the creation of such an instrument, which can reliably and accurately extract features of a patient's gait and aid the clinicians in the assessment of PD.

2.1.2 Treatment

While there is no cure for PD, pharmacological and surgical treatments that alleviate and manage symptoms exist.

The drug levodopa was shown to reduce the disease's symptoms [11]. In more severe cases a subthalamic nucleus deep brain stimulation (STN-DBS) implantation is often used and was shown to significant ameliorate a set of symptoms, particularly some gait disturbances [10].

2.1.3 Gait Disturbances

As mentioned, this thesis focuses on the gait disturbances cause by PD.

These disturbances can be either episodic - where symptoms appear randomly and intermittently and can include start hesitation and freezing of gait, or continuous - which include gait variability, reduced postural control and slow ambulation, caused by bradykinesia and reduced stride length [5].

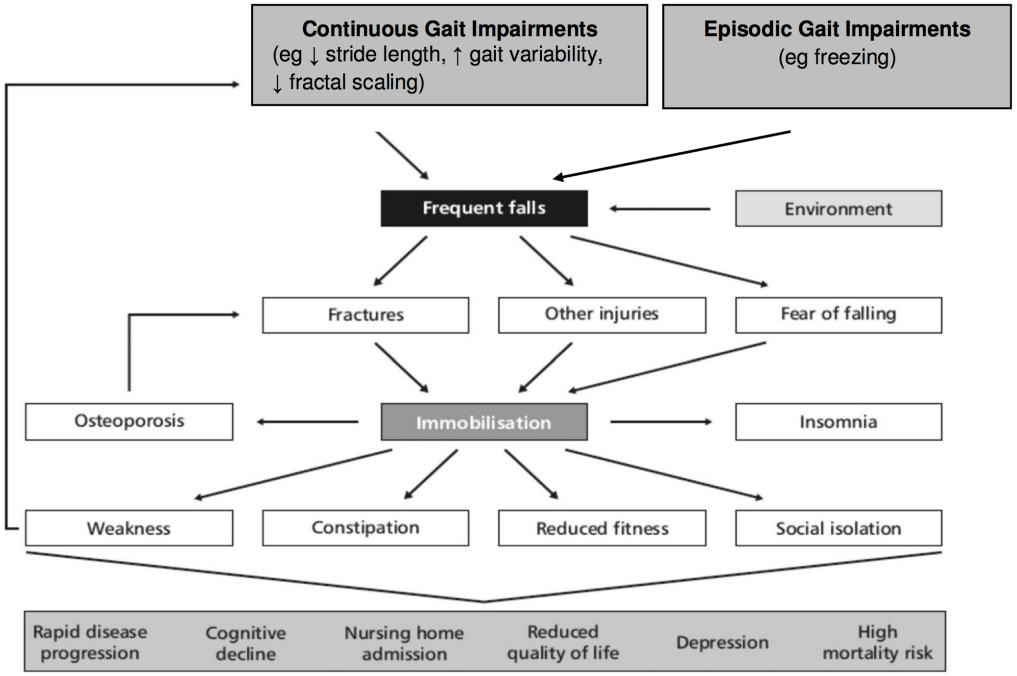


Figure 2.1: Cycle of gait disturbances' impact on a PD patient [5]

Figure 2.1 shows how gait disturbances in PD can greatly impact patients lives, by increasing the risk of falling. The occurrence of falls leads, in turn, to injuries which can provoke immobilization in the patient, which causes further gait impairments. This prompts a decrease in the patient's general well being and can ultimately precipitate death.

2.2 Gait Definition

Human gait is divided into several phases, as described in Figure 2.2

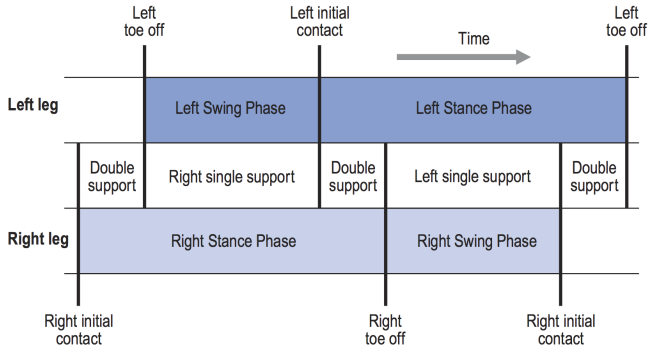


Figure 2.2: Phases of a gait cycle [12]

The two main phases are the swing and stance phase. The swing phase relates to when the foot is not in contact with the ground, while when in the stance phase it is - they are mutually exclusive. They are separated by the toe off event, when the swing is initiated, and the initial contact - or heel strike, when the foot strikes the ground again. This is quite clearly visible in Figure 2.3, where the right leg position is illustrated for each gait event. The stance phase is usually longer, taking about 60% of a gait cycle. It is also denominated as the support phase, which is how it will be referred throughout this thesis. The double support phases relate to when both feet are in contact with the ground while the single support signifies that only one is.

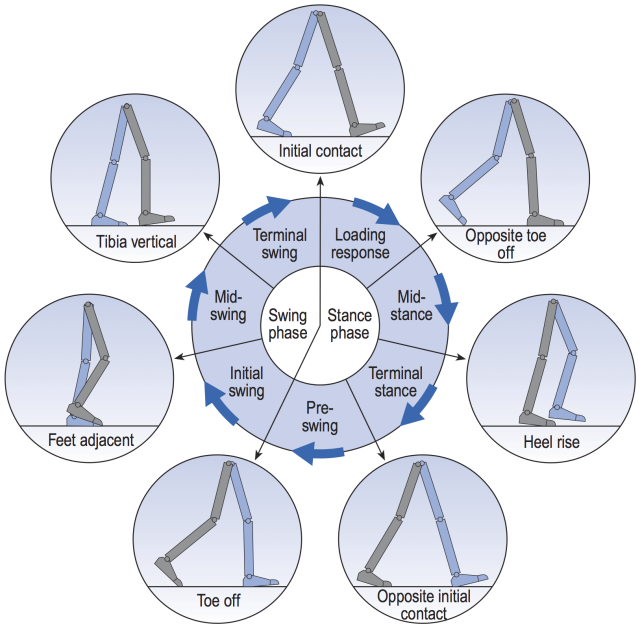


Figure 2.3: Positions of the right leg (in gray) during a full gait cycle [12]

A gait cycle is defined as the period of time between the an occurrence of a gait event and its repetition (e.g., the time between a right leg toe off and the next right leg toe off) [12].

Three relevant gait features are described in [12]: cadence, cycle time and speed. Cadence is usually expressed in steps per minute and pertains to the number of steps walked in a period of time. The cycle time is also used in alternative and is defined as $120/cadence\ (s)$. Speed is also defined as $stride\ length\ (m)/cycle\ time\ (s)$, where stride length is "the distance between two successive placements of the same foot" [12].

2.3 Quantitative Gait Assessment Methods

Quantitative gait analysis methods can be useful in measuring symptoms, such as gait variability and asymmetry that are not easily visible or detected in a normal clinical observation [5]. Several types of quantitative methods and instruments for gait feature extraction have been studied and developed.

In [13], an extensive analysis on methods for gait event detections in ambulatory settings was performed. Different types of measurement methods are identified, such as force based instruments which include insole pressure sensitive sensors and inertial instruments such as the accelerometer, gyroscope and magnetometer based sensors.

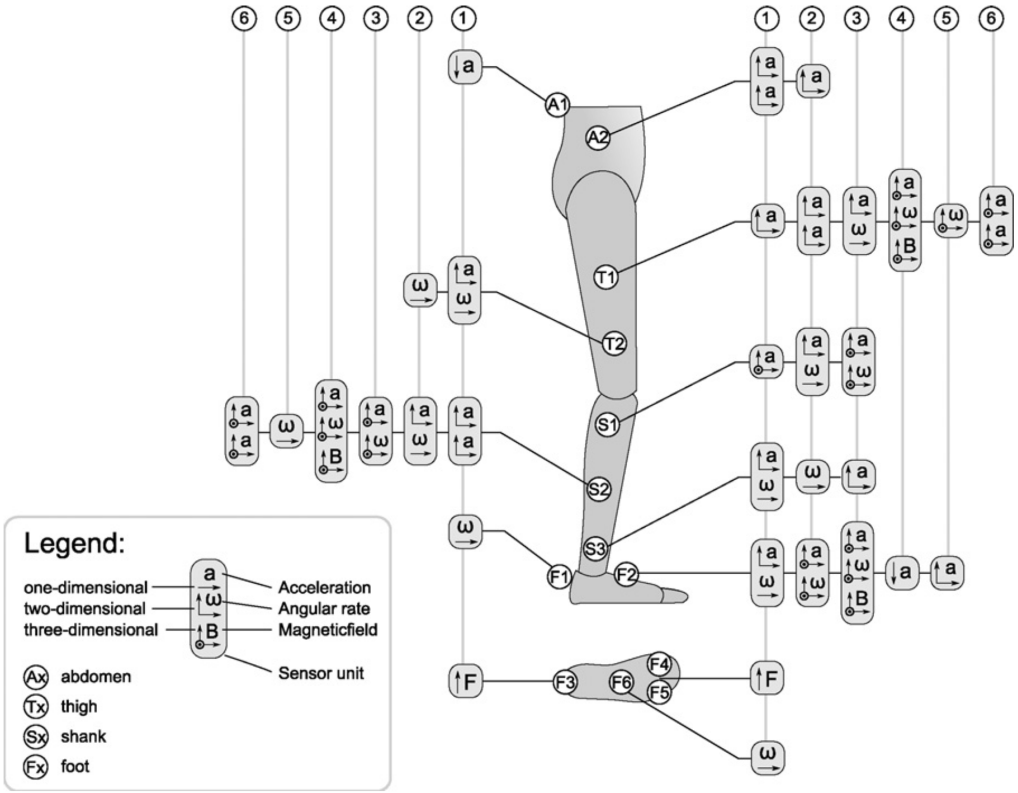


Figure 2.4: Placement of sensors in different lower body segments [13]

2.3.1 Ambulatory Systems

Episodic manifestations of PD, such as FoG episodes, are hard to detect in clinical environments, although its assessment and treatment can be critical to the patient. Furthermore, the awareness of being observed can modify normal behaviors. As such, there is a need for systems that are capable of monitoring a patient's gait in an ambulatory setting, preferably without interfering with a patient's activities [14].

Another aspect that warrants the necessity for ambulatory systems is the ability to perform analysis throughout a wide time-span which is not practical to perform in a clinical setting. For example, fractal analysis of PD gait as described by Hausdorff [5] which reflect "long-range correlation in stride times" throughout hundreds of strides can only be obtained through large samples.

2.3.2 Inertial Systems

The advent of Micro-Electro-Mechanical Systems (MEMS) enabled the development of small and inexpensive IMUs, making them ideal to perform gait analysis [13].

Methods for PD evaluation using IMUs [9, 15], and aimed towards an ambulatory setting [10] have already been developed but aren't yet part of standard clinical evaluations, although this trend is expected to change in the near future [14]. Furthermore, IMUs have been demonstrated to be a reliable tool in performing gait analysis in PD patients [15].

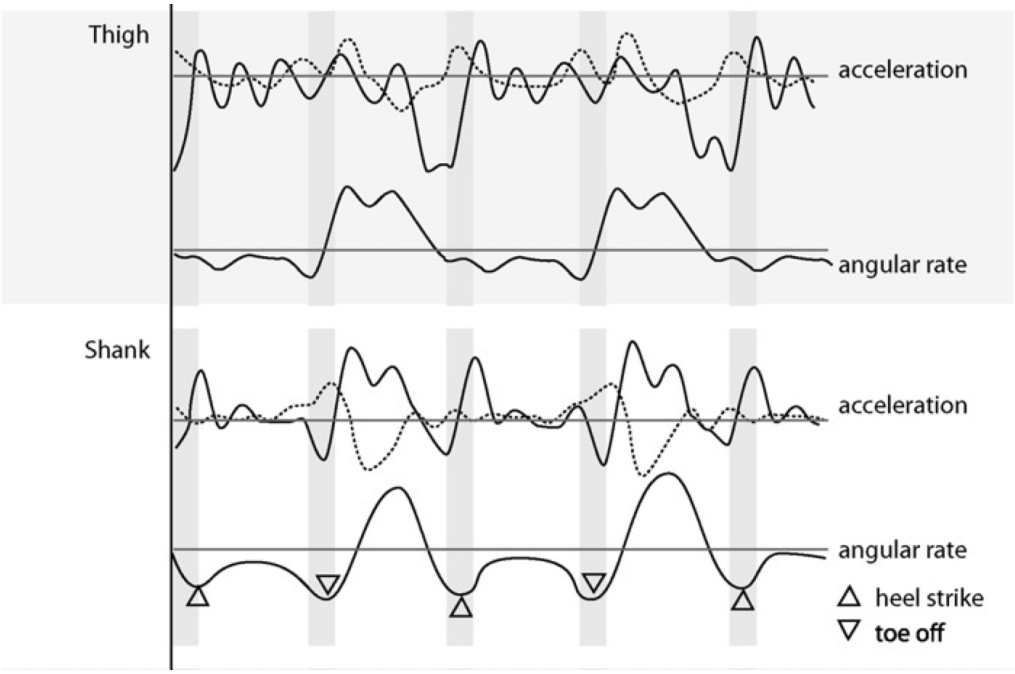


Figure 2.5: Placement of sensors in different lower body segments [13]

IMUs carry several advantages over optical systems. Namely they are quite inexpensive and can be used in an ambulatory setting, throughout a patient's daily activities. They are also capable

of providing spatio-temporal data unlike insole pressure sensors which only provide temporal data [16].

Figure 2.4 shows the position of inertial and pressure sensors throughout the studies reviewed in [13]. Three studies in particular [17, 18, 19] utilized triaxial gyroscope, accelerometer and magnetometer equipped sensors both in the thigh and in the shank, similarly to the settings employed in Chapter 8. One of those studies obtained a 2,6% average error in stride length estimation when compared to a Vicon optical measurement system [19].

Figure 2.5 shows the result of the combination of several studies that placed accelerometers and gyroscopes both in subjects' thighs and shanks. These results are noticeably similar to the ones obtained in Chapter 7, particularly regarding the shank's gyroscope and accelerometer analysis.

The validity of these systems has been widely demonstrated. Furthermore, accelerometers were shown to be the most widely used for gait analysis, often in combination with gyroscopes. It was also reported that sensor position through the thigh, shank or other lower body positions is not critical as correction is possible with signal processing [13].

2.3.3 Optical Systems

Optical motion capture systems are frequently used to in kinematic studies. They are usually based on the spatio-temporal recording of markers placed in a subject's body, which can often get occluded. They are also expensive and confined to the space they are set up [20].

Other markerless systems have been developed. In [21], Microsoft's Kinect RGB-D camera is presented as a novel way to perform PD assessment, providing a much lower cost solution when compared to other marker mocap systems, and also carries the advantage of being a markerless system. It has however a limited range, and is not suitable for ambulatory evaluations.

Chapter 3

Proposed Solution

In this chapter, a description of proposed solution is provided.

The NeuroGait system comprises the IMUs and the smartphone device with a developed Android application. It was designed with PD evaluation as its main goal, however the system is potentially useful for other gait assessment applications as well.

3.1 System Architecture

The system is comprised by a set of Motion Motes, or MoMos, described in detail in Chapter 4, which are paired to an Android phone through a Bluetooth connection.

While MoMos serve the purpose of generating and streaming data, the Android device captures the MoMos' data, acts as a temporary storage unit and enables video to be recorded synchronously. A computer must also be used to permanently store the data and perform all the gait analysis and feature extraction.

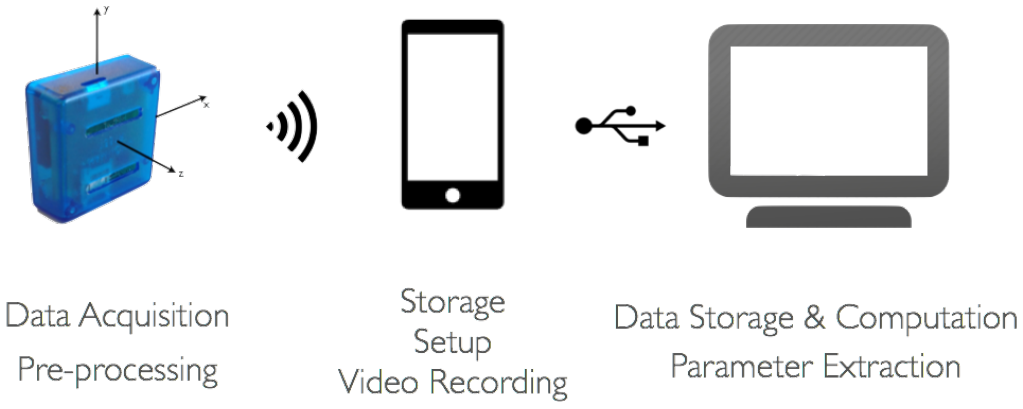


Figure 3.1: Diagram of system architecture

Figure 3.1 depicts a simplified representation of the system architecture. MoMos stream data directly through a previously established Bluetooth connection.

The app then records data by the user's command. These data are stored in the Android device's internal memory. The data can be retrieved by simply plugging the device into a computer via a USB connection.

While the communication between the MoMos and mobile device is performed in real-time, the connection with the computer is established at a later stage, after each trial has been performed.

3.2 Features

The proposed system is easy to set up and use and therefore has the potential to be used not only by clinicians but also by patients in an ambulatory setting.

The MoMos' long battery life, coupled with the Android device's battery life and possibility of being recharged without interrupting its operation, capacitate the system with the ability of recording very long gait datasets.

As an Android device is used, the app has the capability of recording video simultaneously with the MoMo stream, which seems to be an unique feature in inertial sensor based gait capture systems.

Chapter 4

Hardware Analysis

In this chapter, a thorough description the MoMos is presented. Several tests were also performed in order to accurately determine the sampling frequency and consistency between uses.

4.1 Hardware description

Figure 4.1: MoMo with a 1 Euro coin for scale

The Motion Motes, or MoMos, are Bluetooth-enabled Inertial Measurement Units (IMUs) developed at the Institute of Electronics and Telematics Engineering of Aveiro (IEETA). These devices are able to capture data from the triaxial accelerometer, gyroscope and magnetometer sensors. MoMos are battery-powered and are able to stream data continuously up to a full day on a single charge.

These devices can stream raw sensor data and can also compute quaternions[1] in real-time.

Figure 4.2 illustrates the components of the device and how they interact with each other.

The micro-controller gathers data from the sensors at a frequency of about 42 Hz, which then builds packets containing the data and relay them to the Bluetooth module to be transmitted to a synced device.

[1] More on quaternions at http://mathworld.wolfram.com/Quaternion.html

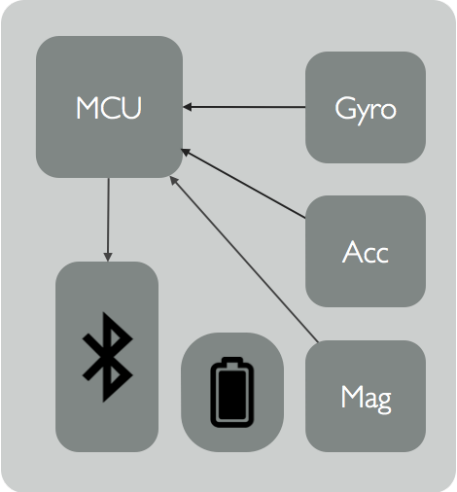


Figure 4.2: Representation of the MoMo's architecture

The sensors used in the MoMos are the following:

- Gyroscope: Invensense's ITG-3200 gyroscope samples data at a 16 bit resolution with a range of ±2000º/s and has a 6.5 mA operating current [22].
- Accelerometer: The KXTF9-1026 by Kionix has user-selectable ranges of 2 g, 4 g and 8 g, can sample with an 8-bit or 12-bit resolution, and draws between 600-800 µA depending on the selected resolution [23].
- Magnetometer: Honeywell's HMC5883L has a minimum range of $\pm 1,0 \times 10^{-4} T$ and a maximum of $\pm 8 \times 10^{-4} T$. With a 12-bit resolution, it is capable of a compass heading accuracy of 1º to 2º. It has a power consumption of 100 µA [24].

These devices are also relatively small-sized, as can be seen on Figure 4.1, so the disturbance to the gait when worn by a subject is minimum.

4.1.1 Axis Orientation

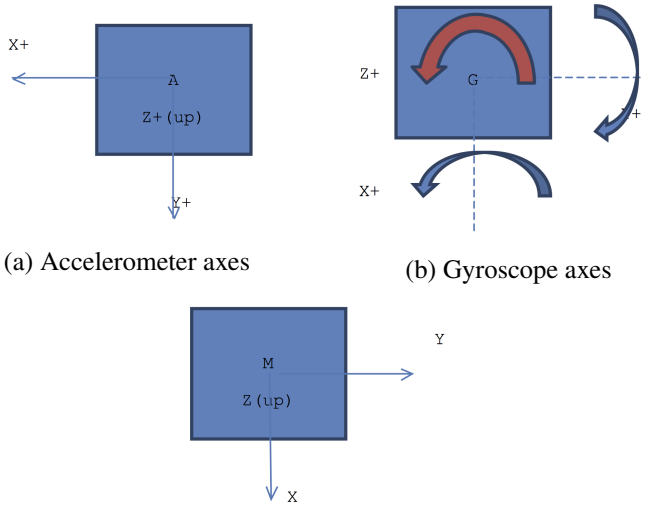
Figure 4.3 details the orientation of the MoMos axes for each of the three sensors. Throughout this thesis, all the trials were performed with the accelerometer's y-axis facing upwards.

Figure 4.4 shows the accelerometer's axes overlaying the device.

4.2 Operation

4.2.1 Operation Modes

The MoMo has three different operating modes, two for streaming data either in quaternions or raw sensor data and one for calibrating the sensors. Every time it is turned on, one of the modes must be selected by the paired device.



(a) Accelerometer axes

(b) Gyroscope axes

(c) Magnetometer axis

Figure 4.3: Orientation of the sensors' axes [25]

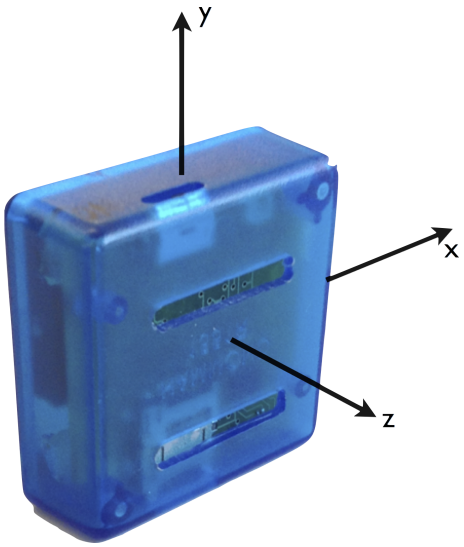


Figure 4.4: Photograph of MoMo with accelerometer axes represented

If the one of the streaming data modes is selected, the device will start to broadcast data in a few seconds and a yellow LED will blink. If the calibration mode is selected, the paired device will receive a message announcing when the calibration has begun and once the procedure is finished it will return offset data.

4.2.2 Workflow

In this section, the typical usage of the MoMos is described.

Figure 4.5 details how the MoMo functions. Following, each action from the activity diagram will be detailed.

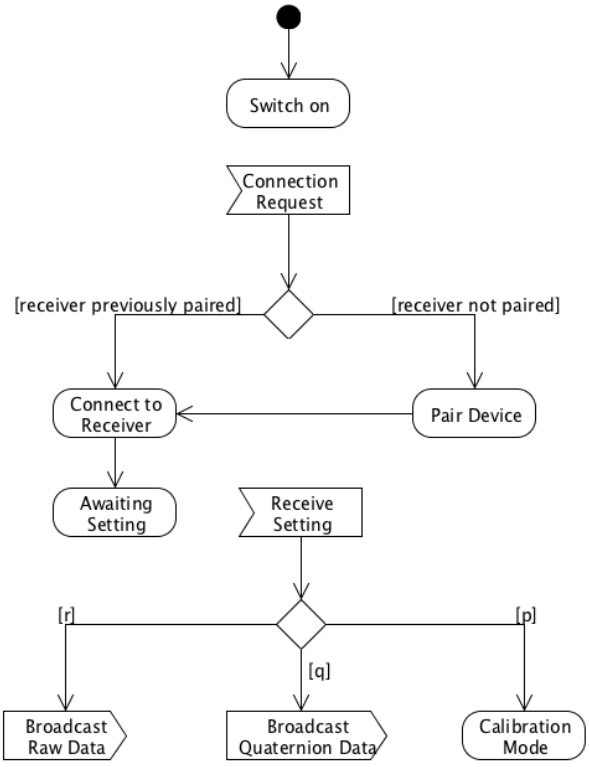


Figure 4.5: Activity diagram for the MoMo's workflow

- Switch On: the MoMo has a physical on/off switch, which must be flipped on in order to start the device.
- Connection Request: a Bluetooth-enabled device, which will receive the data, must discover the MoMo's network and attempt to connect to it.
- Pair Device: If the MoMo hasn't connected with the device before, it must be paired using '0000' as the code.
- Connect to Receiver: upon a connection request, the MoMo establishes a link with the device.
- Awaiting Setting: once the connection is established, the MoMo waits for a message from the device which will define its mode of operation.
- Receive Setting: the MoMo receives from the device one of three characters. Either 'r' for raw data, 'q' for quaternion data or 'p' to calibrate the device.
- Broadcast Raw Data: the MoMo continuously streams raw data from all three sensors.
- Broadcast Quaternion Data: the MoMo computes quaternions from the sensors and broadcasts the data.
- Calibration Mode: the MoMo initiates a calibration procedure.

Once one of the broadcast modes is selected the device will only stop streaming data if it is switched off or runs out of battery.

4.3 Data Packets

As described above, MoMos are able to stream raw sensor or quaternions data. This section details how they package the data and how the sensors are set up.

4.3.1 Range and Resolution Setup

The MoMo's firmware configures the sensors' output. The set of four MoMos used during the development of this thesis were configured by IEETA's team as follows:

- Gyroscope: it is set to a 16-bit resolution with a range of ±2000º/s, meaning it outputs values between -32768 and 32767.
- Accelerometer: this sensor is configured to have an 8-bit resolution ranging from -2.000 g to +1.984 g [23]. It produces values between -128 and 127.
- Magnetometer: the range is configured $\pm 1,0 \times 10^{-4} T$. Its fixed 12-bit resolution outputs values between -2048 and 2047.

The resolution on each sensor details how many steps there are between their range limits.

4.3.2 Packet Structure

Table 4.1 details the structure of raw data MoMos build using the sensor readings. The first row labels each field. The second row relates the triaxial coordinates to the corresponding sensor. The third row indicates the output resolution from each sensor and the last line specifies the range for each sensor and the *sequence number*.

ID	x	y	z	x	y	z	x	y	z	SEQ
	Gyro			Acc			Mag			
	16 bits			8 bits			12 bits			
	±2000º/s			±2 g			$\pm 1,0 \times 10^{-4} T$			0-999

Table 4.1: Raw data packet structure

The device starts its packets by sending an identifier, usually a letter and a number (e.g. Q1). It then writes the raw data of each axis of each sensor, as detailed in Table 4.1. The packet finishes with a sequence number.

The sequence number is a number between 0 and 999 and is incremented by one with each sent packet. The sequence number as not initially a feature in the MoMos; however after some discussion with IEETA's development team, it was requested that it was added in order to help

determine if a transmission error had occurred. The reasoning for this was that it would be simple to programmatically detect if a package didn't arrive successfully as it would cause an evident gap between one packet and another. It would also help to reorder the packets if some did not arrive sequentially.

All fields in each packet are separated by a comma, as presented in Listing 4.1.

```
Q6,-98,-3445,-111,132,64,127,-232,140,363,581
```

Listing 4.1: Packet example

4.3.3 Value Conversion

MoMos return numerical values from the sensor readings as detailed in 4.3.1. In order to convert them to physical units, simple mathematical operations must be performed.

For the acceleration a, the value x is divided by 127 and multiplied by the earth's gravity g, as described in 4.1.

$$a = \frac{x - 127}{64} g \quad (\mathrm{m\,s^{-2}}) \tag{4.1}$$

The angular velocity ω is similarly obtained:

$$\omega = \frac{x}{32767} 2000 \quad (^{\circ}\,\mathrm{s^{-1}}) \tag{4.2}$$

and the magnetic flux density B from the magnetometer:

$$B = \frac{x}{2047} 10^{-4} \quad (T) \tag{4.3}$$

4.4 MoMos Frequency Assessment

In order to accurately determine the packet output frequency of each one of the MoMos provided, a test was devised.

Four MoMos were placed on a table and their data stream along with video was recorded using the Android application. While recording, the table was knocked periodically in order to cause a visible change in the data stream. To time each event, a GoPro Hero3+ camera was set to record at 100 frames per second. This allows the MoMos' signal, which were known to be below 50 Hz, to be oversampled. Avidemux 2.6[2] was then used to perform a frame-by-frame video analysis.

To calculate the MoMos output frequency, the number of packets received for each MoMo within two knocks separated by 64,89 seconds was counted.

Table 4.2 shows the number of packets between each knock and the calculated frequency for each MoMo. They are identified by their packet identifier (see Section 4.3.2).

[2] Avidemux is a video editor available under GNU GPL license. More on http://fixounet.free.fr/avidemux/

MoMo ID	Packets	Frequency (Hz)
Q1	2759	42.52
Q2	2818	43.43
Q5	2759	42.52
Q6	2822	43.49

Table 4.2: Frequencies for each MoMo

The MoMos Q1 and Q5 output the same frequency. However, Q2 and Q6 have different frequencies, which disallows packet by packet comparison between MoMos for datasets larger than a few seconds.

A second set of tests were performed, using the same procedure although using a much larger time period - 10 and 11 minutes. Although these showed similar values to the ones measured previously, the calculated values for each MoMo were significantly different from each other, suggesting that these devices may slightly vary their sampling frequency, possibly due to temperature variations between tests within its components.

4.5 Conclusions

The MoMos are a key component in the proposed system. They are lightweight durable devices that can easily be fitted in any position in the body and have a long battery life which makes them suitable for long periods of use in an ambulatory setting.

They were found to be quite reliable, although testing revealed, besides differences in sampling frequencies between the four supplied devices, some slight variation in its sampling frequency between tests. Their simple usage and Bluetooth connection make them ideal to link to a mobile device.

Chapter 5

Mobile Application

A mobile application (or app) for Android was developed in order to simultaneously gather data from multiple sensors, record video and store those data for later relaying to a computer. In this chapter, the application's features, functions and usage is thoroughly detailed.

5.1 Requirements

The app was designed several key requirements in mind, namely:

- **Synchronous data collection**: MoMos' data are stored synchronously, i.e., by pressing the Record button, the app will simultaneously start recording data from all the connected MoMos.
- **Video recording**: the user may choose to record video at a resolution of 720x1280 pixels at 25 frames per second. It records the video vertically as it was deemed more suitable to capture gait when compared to recording horizontally. The video is also synchronized with the MoMos' data, meaning the app starts recording both video and MoMo data simultaneously.
- **Data storage**: the app is able to store the data received from the MoMos. For each recording, a folder is created and named with the current date and time. For each MoMo, the app creates a single text file that contains all the data stream.
- **Data retrieval**: stored data can be retrieved by connecting a USB cable to a computer.
- **Body location**: the user can associate a connected MoMo to a location in the subject's body, e.g., the lower right leg. This location will be present in the data recording filename.
- **Simple usage**: the app may be easily and intuitively utilized, with as few steps as possible. It contains a simple user interface that depicts the human body and has buttons for each body position where the MoMos can be applied.

Figure 5.1 shows the use cases for the application and its interaction with users and MoMos.

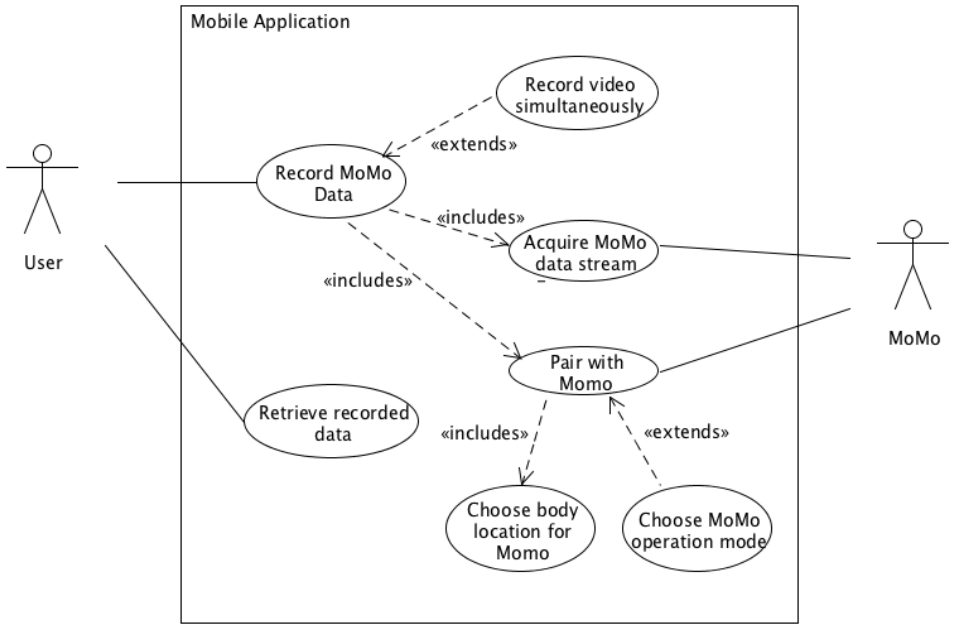


Figure 5.1: Use case diagram for the mobile application

5.2 Application Workflow

In this section the typical workflow of the mobile application is described, from the initial setup and connection to the MoMos to the actual data recording.

The activity diagram of Figure 5.2 represents the activities performed in a normal usage of the application. This diagram focuses on the general actions that are triggered by the user and which generate feedback within the user interface. It does not go into detail on what occurs programatically. This will be discussed in a further section.

Following, the app usage will be described through an example of the typical utilization with the aid of screenshots of the app's UI. Each of the actions described in the activity diagram will be mentioned as the usage is delineated.

Naturally, the user starts by pressing the app icon in the Android's interface (**Start App**). On the main screen, a depiction of a human body is presented along with buttons that represent each of a possible body location, as can be seen on Figure 5.3 (**Select Body Location**). The placement of these buttons was partly selected based on Figure 2.4. If a MoMo had already been connected and associated to that body location, the app would terminate the connection with that device (**Disconnect from MoMo**) and free that location, returning the button to an 'off' state, i.e., not highlighted (**Free Body Location**).

The user then proceeds to select a body position where a chosen MoMo has been or will be placed. Here, the user selects the right lower arm.

The app then presents a list of previously synced Bluetooth devices and actively searches for new ones. Raw data mode operation is selected by default but the user may opt to choose another

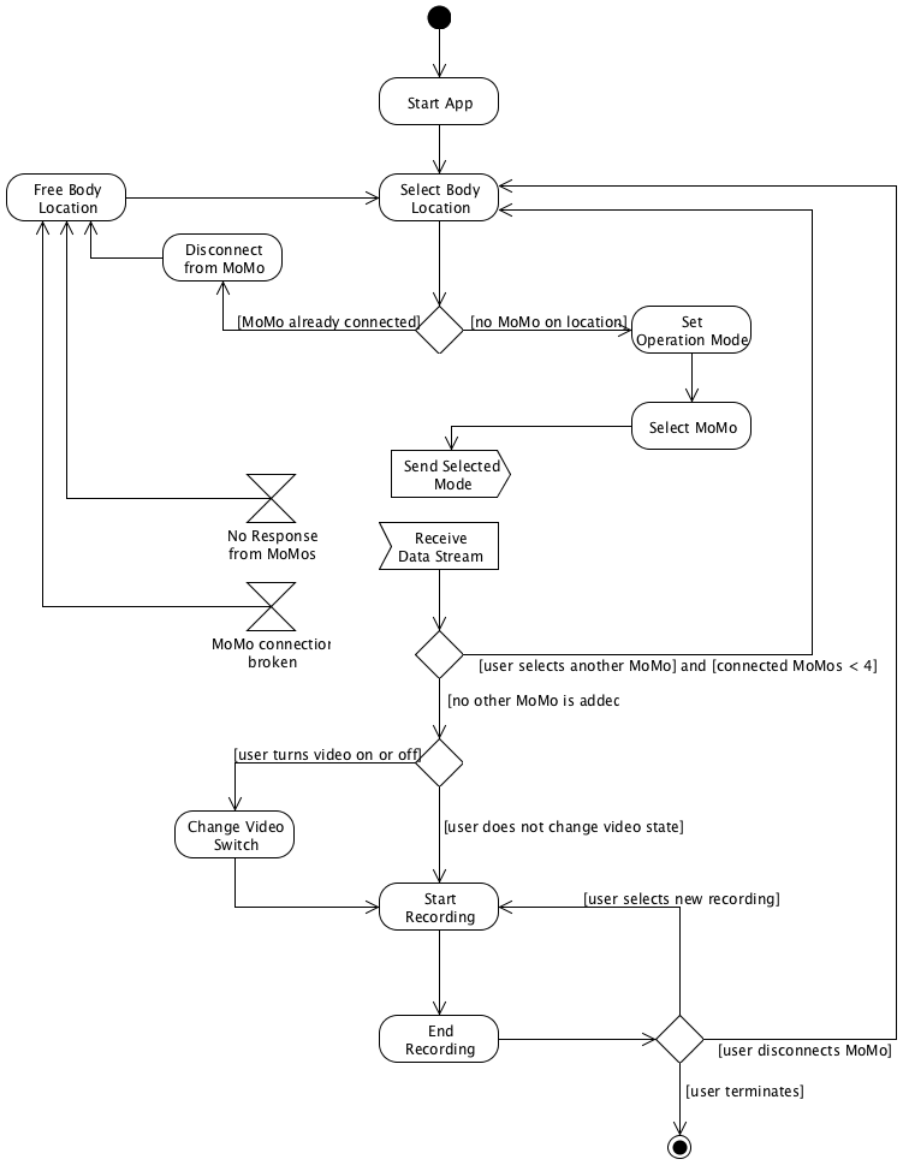


Figure 5.2: Application Main Workflow Activity Diagram

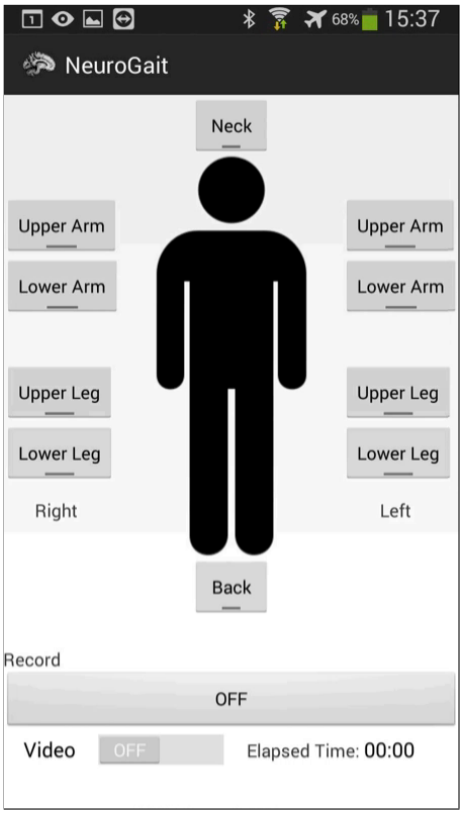


Figure 5.3: Application main screen

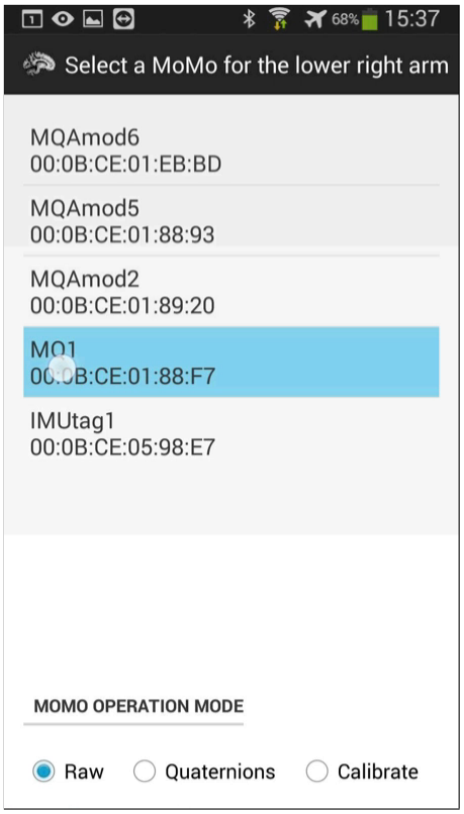


Figure 5.4: Selection of MoMo

before selecting a MoMo (**Set Operation Mode**). To proceed a MoMo must be selected. On Figure 5.4, the user selects the MoMo with a device name of 'MQ1'. Afterwards, the app returns to the main interface. If the connection is successful, a message will pop-up notifying that event, as can be seen on 5.5. The app will start receiving data from the IMUs as soon as the MoMo starts blinking, although at this point it will not record. If no connection is established after a few seconds the app will notify the user the connection attempt has failed and will release the selected body position (**No Response from MoMos**). The user may continue to connect more devices, up to a maximum of four streaming simultaneously.

After successfully pairing the devices the user is ready to start recording. Video may be enabled by switching on the video button (**Change Video Switch**), as depicted on Figure 5.6. Recording starts by pressing the record button (**Start Recording**). If the video is enabled, the user can simply scroll down to view what is being recorded. The app also features an elapsed time counter, letting the user know for how long it has been recording.

Once the user wishes to do so, the recording may be stopped by pressing the record button again (**End Recording**). Users may then record again using the same settings or modify the app's parameters.

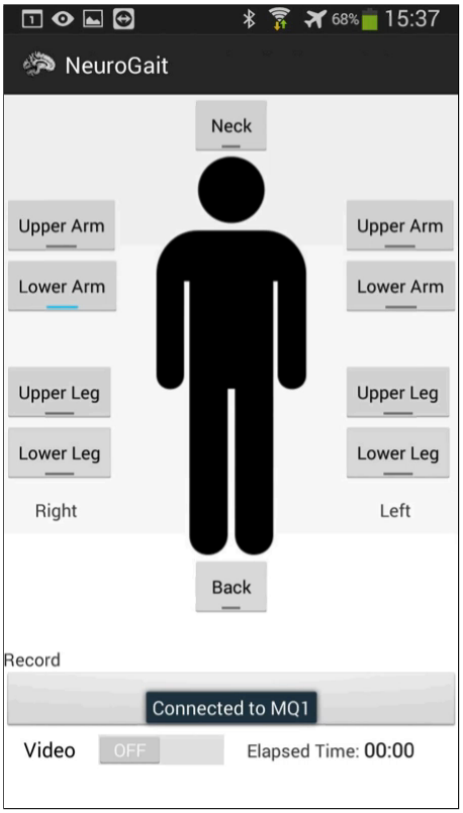


Figure 5.5: MoMo connected

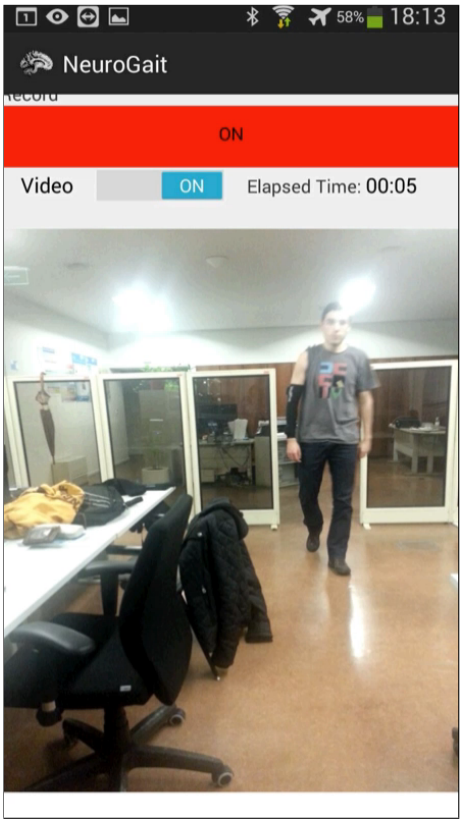


Figure 5.6: Recording in progress

5.3 Application Implementation

The application was developed using the Java programming language [26] under Android Developer Tool [27]. In this section, the development process is described and all the program's main elements are thoroughly detailed.

5.3.1 Development Steps

The development of this application posed several challenges, which were largely overcome. Following, each of the main steps which were part of the implementation of the app are described.

5.3.1.1 Bluetooth Communication

One of the main challenges was the development of a reliable form of simultaneously receiving continuous streams of data from multiple Bluetooth devices. This was not found to be a typical usage of the Bluetooth protocol, specially on mobile equipments, even though Bluetooth supports up to 7 simultaneously connected devices, set as slaves [28].

Firstly, an attempt to establish a simple connection to a MoMo was developed. Using Android's Bluetooth API guide and the Bluetooth Chat Sample app as a base [29], a connection with

a single device was successfully established. MoMos communicate using Bluetooth's Serial Port Profile [29], similarly to what's used in the sample app.

Following the guide, it was fairly straightforward to establish a connection with the MoMos. At this point, the app was capable of receiving data, but not transmitting. In order to set a MoMo operation mode, an app called Bluetooth SPP[1] was used, enabling the smartphone to send commands to the MoMos. Using this setup, it was already possible to receive and visualize MoMo data using only the Android device.

The next step was to enable multiple simultaneous connections. No mention on how to implement this feature was found on Android's official documentation. However, a suggestion[2] led to attempt multiplying the thread instances made to initiate each Bluetooth connection. Therefore, by replicating the process to initiate each Bluetooth connection and allowing Android to instantiate an individual thread for each active connection, the app was enabled to receive multiple simultaneous connections. This process of connecting to a device is described in detail on Section 5.3.2.

5.3.1.2 Video Recording

The implementation of the video recording in the application was entirely based on Android's Camera API Guide [30]. This feature is easily accessed by switching on the Video button in the main UI. The video feed is available within the UI, simply by scrolling down to the camera view. The camera was set to record with a resolution of 720x1280 pixels at 25 frames-per-second.

The video is set to record vertically. The reasoning behind this is that, as the main focus of the application is to record gait, recording vertically would better suit the capturing of the full human figure while walking.

5.3.2 Classes Overview

The application is organized in four main files, which closely relate to its main classes. On Figure 5.7, the app's main classes are presented. The figure includes all of each class's methods but not its attributes.

The classes `MainActivity` and `SensorListActivity` closely relate to the two main screens of the UI, while `BluetoothCommService` and `CameraPreview` have the methods necessary to establish and manage Bluetooth connections and obtain a video feed from the camera, respectively.

Following, each class and its main methods are described.

5.3.2.1 MainActivity Class

The `MainActivity` class is the app's main class as it implements most of the methods of the app's workflow and is responsible for handling the events in the UI's main screen.

[1] Android application available at https://play.google.com/store/apps/details?id=mobi.dzs.android.BluetoothSPP

[2] Suggestion provided by a Stack Overflow user at https://stackoverflow.com/questions/7053804/dual-spp-bluetooth-connexion-on-android

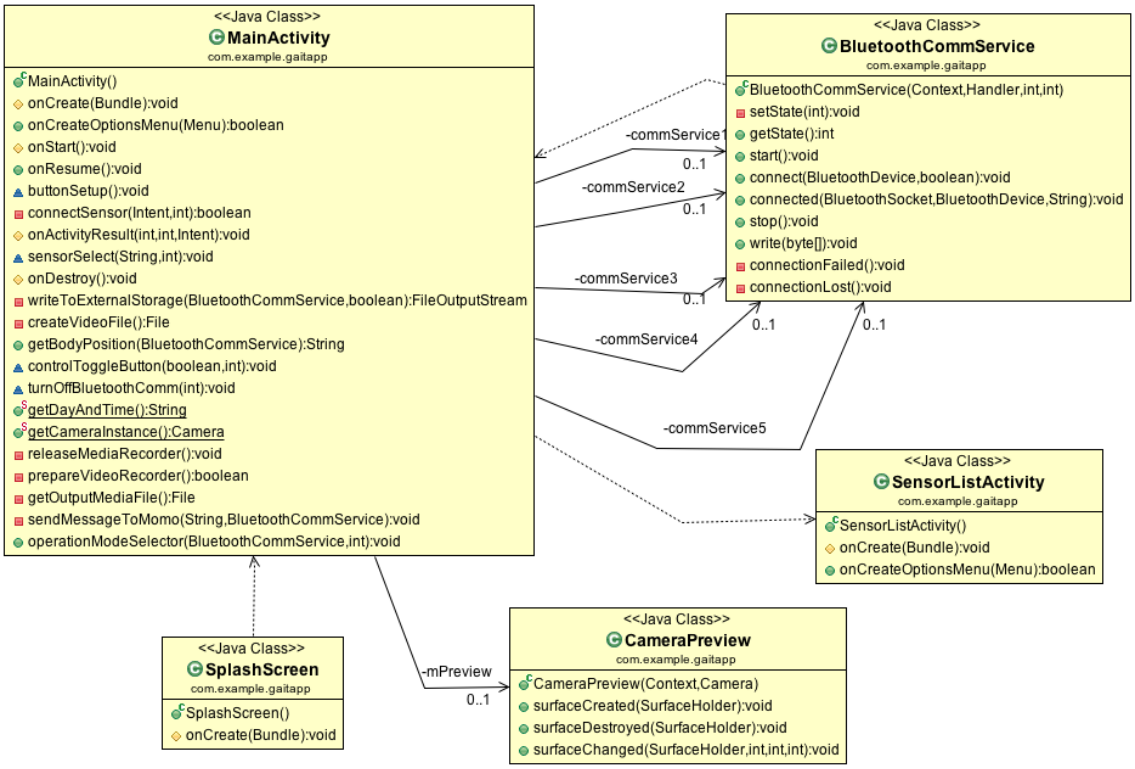


Figure 5.7: Application Class Diagram

Besides the customization of Android's own methods such as `onCreate()`, `onStart()` or `onResume()` which are called in response to system events such as the app being initiated or coming from a resumed stated, the methods implemented in this class process most events throughout the app.

Ensuingly, each of these methods is described. They are listed in the typical order by which they are called on a normal usage of the app.

`buttonSetup()`

This method is called by `onResume()` and instantiates the necessary methods for each button of the main UI - the buttons for each body position and for enabling video capturing and to initiate a recording.

```
1 lowerRightArmTB = (ToggleButton)findViewById(R.id.lowerRightArmToggleButton);
2 lowerRightArmTB.setOnCheckedChangeListener(new CompoundButton.
      OnCheckedChangeListener() {
3
4     @Override
5     public void onCheckedChanged(CompoundButton buttonView, boolean isChecked
      ) {
6         if (isChecked){
7             sensorSelect("lower right arm", REQUEST_LOWER_RIGHT_ARM_SENSOR);
8         }
9         if(!isChecked){
```

```
            try {
                if(!recording)
                    turnOffBluetoothComm(REQUEST_LOWER_RIGHT_ARM_SENSOR);
                else
                    lowerRightArmTB.setChecked(true);
            } catch (Exception e) {}

        } // trying to disconnect the momo if the user chooses to. only if not recording
    }
});
```

Listing 5.1: Setup of the button for the lower right arm

Listing 5.1 shows one of the buttons declaration and respective methods. The method `setOnCheckedChangeListener`[3] listens for press events on the Toggle Buttons. Whenever a button press occurs, if the button gets highlighted (or checked) the method calls `sensorSelect` which will trigger the MoMo selection process, else it will attempt do disconnect the device from the MoMo and free that Bluetooth connection so it can be reused. It will not allow to disconnect if a recording is in progress, i.e. pressing the button will not yield any effect.

`sensorSelect(String, int)`

This method receives a body position set by the Toggle Button that called it and initiates an Intent[4] to start the `SensorListActivity`.

```
Intent serverIntentDev = new Intent(MainActivity.this, SensorListActivity.class);
serverIntentDev.putExtra("bodyPosition", bodyPosition); // sends string to fill the title in the following activity
startActivityForResult(serverIntentDev, BODY_POSITION_CODE);
```

Listing 5.2: `sensorSelect(String, int)` method

It is also responsible for getting back the data from that activity, namely the MoMo and settings selected, which in turn is gathered by the `onActivityResult` method.

`onActivityResult(int, int, Intent)`

This method is called by the OS, after `SensorListActivity` returns. Its purpose is to call out the `connectSensor()` method with the correct data using a switch statement[5], as exampled in Listing 5.3.

[3] Android documentation for Toggle Buttons available at https://developer.android.com/guide/topics/ui/controls/togglebutton.html

[4] On Android an Intent launches another activity. More at https://developer.android.com/reference/android/content/Intent.html

[5] More about the Java switch statement at http://docs.oracle.com/javase/tutorial/java/nutsandbolts/switch.html

```
switch (requestCode) {
case REQUEST_UPPER_RIGHT_ARM_SENSOR:
    if (resultCode == Activity.RESULT_OK) {
        connectSensor(data, requestCode);
    }
    break;
```

Listing 5.3: Excerpt of the switch statement of `onActivityResult`

`connectSensor(Intent, int)`

This method is responsible for initiating the required procedures to connect to a MoMo. It creates an instance of the `BluetoothDevice`[6] class using the MoMo's MAC address obtained from of `SensorListActivity`. It then instantiates the `BluetoothCommService` class and calls the connect procedure with the selected MoMo, as detailed in Listing 5.4.

```
private boolean connectSensor(Intent data, int requestCode) {
  // Get the device MAC address
  String address = data.getExtras()
      .getString(SensorListActivity.EXTRA_DEVICE_ADDRESS);
  // Get the BluetoothDevice object
  BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
  // Attempt to connect to the device
  if(DEBUG) Log.e(TAG, "Trying to start connection");
  int operationMode = data.getExtras().getInt(SensorListActivity.
    SELECTED_RADIO_BUTTON);
  if(DEBUG) Log.e(TAG, " Operation mode is: " + operationMode);

  if(commService1 == null){
      if(DEBUG) Log.i(TAG, "Connecting using commService1");
      commService1 = new BluetoothCommService(this, mHandler1, requestCode,
    operationMode);
      commService1.connect(device, false);
```

Listing 5.4: Part of the `connectSensor` method

`recordButton`

While not a method but instead a `ToggleButton` field, it is important to explain what was implemented on this particular button, as it handles how the recordings proceed. The function that was mentioned on the `buttonSetup()` method description also controls behavior here in the same manner.

```
public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
    if (isChecked){
        recording = true;
        recordButton.setBackgroundColor(Color.RED);
```

[6] More on this class available at https://developer.android.com/reference/android/bluetooth/BluetoothDevice.html

```
5           videoTimer.setBase(SystemClock.elapsedRealtime());
6           videoTimer.start();
7
8           if(videoSwitch.isChecked()){
9               if (prepareVideoRecorder()) {
10                  // Camera is available and unlocked, MediaRecorder is
    prepared,
11                  // now you can start recording
12                  mMediaRecorder.start();
13                  }
14              }
```

Listing 5.5: Start of recording

As the code on Listing 5.5 shows, when the button is pressed in order to initiate a recording, the app first sets a boolean flag named `recording` to `True`. This is critical, as it indicates the handler threads managing the Bluetooth communications that a recording has initiated or is in progress and that the data stream they are receiving must be saved. Next, a timer is initiated in order to show the user the elapsed time during the recording. Then, the camera switch is checked. If it is on and the camera is correctly set up, it immediately starts saving the video feed.

```
1  if(!isChecked){
2      stop = true;
3      recordButton.setBackgroundColor(Color.LTGRAY);
4      recording = false;
5
6      if(videoSwitch.isChecked()){
7          mMediaRecorder.stop();  // stop the recording
8          releaseMediaRecorder(); // release the MediaRecorder object
9          mCamera.lock();         // take camera access back from MediaRecorder
10     }
11     videoTimer.stop();
12 }
```

Listing 5.6: Stopping the recording

When the record button is pressed while a recording is in progress, the code in Listing 5.6 is executed. It sets the `stop` boolean flag to `True` and `recording` to `False` to signal that not only the recording is not in progress but it also has been effectively stopped. If the video had been on, it would have proceeded to stop the recording and release the camera. This part of the procedure was also based on the previously mentioned Camera API Guide.

`handleMessage(Message)`

Each instance of the `BluetoothCommService` class has an instance of a `Handler`[7].

[7] Android Handlers are further described at https://developer.android.com/reference/android/os/Handler.html

Each `Handler` instance is associated with a connection to a Bluetooth device. It is responsible to receive the messages sent by the corresponding `BluetoothCommService` objects. The `handleMessage` method receives all the state changes and messages back from the thread that handles each MoMo connection and sends messages on command. It implements how each MoMo instance manages the received messages and how it passes operation mode commands back to the MoMos. This method was based on the one with the same name implemented on BluetoothChat example but was heavily adapted in order to fit the designed specifications. It mainly consists of a switch statement that handles each call from the device.

```
1   public void handleMessage(Message msg) {
2       switch (msg.what) {
3       case MESSAGE_STATE_CHANGE:
4           if(DEBUG) Log.i(TAG, "MESSAGE_STATE_CHANGE: " + msg.arg1);
5           switch (msg.arg1) {
6           case BluetoothCommService.STATE_CONNECTED:
7             Toast.makeText(getApplicationContext(), "Connected", Toast.
    LENGTH_SHORT);
8             operationModeSelector(commService1, commService1.operationMode)
    ;
9               break;
10          case BluetoothCommService.STATE_CONNECTING:
11            Toast.makeText(getApplicationContext(), "Connecting...", Toast.
    LENGTH_SHORT);
12              break;
13          case BluetoothCommService.STATE_LISTEN:
14          case BluetoothCommService.STATE_NONE:
15              break;
16          case BluetoothCommService.STATE_DISCONNECTED:
17            controlToggleButton(false, commService1.bodyPositionCode);   //
     Tells the function to turn the light off at the TB
18            commService1 = null; // Kills the commService so it can be
    reused
19            break;
20          }
21          break;
```

Listing 5.7: `handleMessage` implementation of Bluetooth state changes

Listing 5.7 details how the method handles the `MESSAGE_STATE_CHANGE` case, which in turn executes an action depending on which state the Bluetooth communication has changed to. Specifically, it can: let the application know if the MoMo has just been connected using the `BluetoothCommService.STATE_CONNECTED` flag, in which case it lets the user know with a Toast message[8] that it has been connected and calls out `operationModeSelector` in order to initiate the process that selects the MoMo's operation mode; `BluetoothCommService.STATE_CONNECTING` lets the user know that

[8] More on Toasts at https://developer.android.com/guide/topics/ui/notifiers/toasts.html

the app has initiated the process of connecting to a device; and `BluetoothCommService.STATE_DISCONNECTED` informs the `MainActivity` that the connection to the device has been lost, switches off the Toggle Button linked to that device and nulls that `BluetoothCommService` instance so it can be reused.

```
case MESSAGE_READ:
    byte[] readBuf = (byte[]) msg.obj;
    // construct a string from the valid bytes in the buffer
    String readMessage = new String(readBuf, 0, msg.arg1);
    if(recording){    // if the flag is True the app may record
        if(!file1Created ){ // if the file hasn't been created, it will be
    now
          comm1Stream = writeToExternalStorage(commService1, false);
          file1Created = true;
        if(DEBUG)Log.i(TAG, "file 1 created");
        }

        try {
            comm1Stream.write(readMessage.getBytes());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    if(stop && !recording){
        try {
            comm1Stream.close();
            file1Created = false;
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    break;
```

Listing 5.8: `handleMessage` handling of received messages

The next case relates to the handling of received messages from the Bluetooth device and it is shown in Listing 5.8. This is a critical case as it is responsible for handling all the data received from the MoMos. In the event of a `MESSAGE_READ` case, `messageHandle` first saves the buffer to a byte array. It then checks if it should be recording. If so, it checks if a file for that recording (in this case using the `file1Created` boolean flag) has already been created. If not, it calls out `writeToExternalStorage` which in turn creates a new `FileOutputStream`[9] instance which outputs the data stream to a file, in this case `comm1Stream`. Each time the method is called out, it updates `comm1Stream` with the latest received data buffer. As soon as the recording is stopped, by checking the `stop` and

[9]More about the FileOutputStream class at https://developer.android.com/reference/java/io/FileOutputStream.html

`recording` flags it immediately closes the stream (i.e. closes the created file) and sets the `file1Created` flag to `False`.

```
case MESSAGE_DEVICE_NAME:
    // save the connected device's name
    mConnectedDeviceName = msg.getData().getString(DEVICE_NAME);
    Toast.makeText(getApplicationContext(), "Connected to "
                   + mConnectedDeviceName, Toast.LENGTH_SHORT).show();
    break;
case MESSAGE_TOAST:
    Toast.makeText(getApplicationContext(), msg.getData().getString(TOAST),
                   Toast.LENGTH_SHORT).show();
    break;
```

Listing 5.9: `handleMessage` device name and Toast messages

Listing 5.9 shows the `DEVICE_NAME` case, that sends a Toast notification with the selected device name as soon as it is connected. `MESSAGE_TOAST` enables the `BluetoothCommService` object to send Toast notifications. `handleMessage` also receives and effectively echoes the messages that are sent out to the Bluetooth devices but the app does not actually make use of that feature.

`operationModeSelector(BluetoothCommService)`

This method receives an object from the `BluetoothCommService` and the selected operation mode from `SensorListActivity`.

```
public void operationModeSelector(BluetoothCommService commService, int
    operationMode){

    switch (operationMode) {
    case SensorListActivity.SELECT_RAW_DATA:
        sendMessageToMomo("r\r\n", commService);
        break;
    case SensorListActivity.SELECT_QUATERNIONS:
        sendMessageToMomo("q\r\n", commService);
        break;
    case SensorListActivity.SELECT_CALIBRATION:
        sendMessageToMomo("p\r\n", commService);
        break;
    default:
        break;
    }
}
```

Listing 5.10: `operationModeSelector()` method

As detailed in Listing 5.10, it calls the `sendMessageToMoMo` method with a String that the MoMo expects to receive in order to be set to the operation method chosen by the user and forwards the `BluetoothCommService` object. It sends a string with the letters 'r', 'q' or 'p' appended with the necessary escape characters '\r\n'.

`sendMessageToMoMo(String, BluetoothCommService)`

This method was adapted from BluetoothChat example. It is responsible for calling out `BluetoothCommService.write()` method with the message it receives in order to send it out to the MoMo. It is only used by the `operationModeSelector()` method.

`writeToExternalStorage(BluetoothCommService, boolean)`

This method creates a file for each MoMo data stream when a recording is initiated. It is called out by the MoMo's connection handler when it reads a message from the MoMo for the first time after the recording is started.

```
1  String bodyPosition = getBodyPosition(commService);
2  File root = android.os.Environment.getExternalStorageDirectory();
3  FileOutputStream f = null;
4  // See
5  // http://stackoverflow.com/questions/3551821/android-write-to-sd-card-
   folder
6  File dir = new File(root.getAbsolutePath() + "/NeuroGait/"+ getDayAndTime
   () +"/");
7  File file;
8  dir.mkdirs();
9  file = new File(dir, bodyPosition + " " + System.currentTimeMillis()+".
   txt");
```

Listing 5.11: Excerpt of `writeToExternalStorage()` method

It extracts a String from the received `BluetoothCommService` object which contains the body position associated to that object. It then gets the device's root folder path, appends NeuroGait and the current date and time and creates a new directory with those elements, except if one has already been created with the same date and time by another `BluetoothCommService` instance. Next, it creates a new text file named with the body position and the current system time in milliseconds, when calling `System.currentTimeMillis()`[10]. This time allows to account for differences in the start of creating files when multiple MoMos are connected, and can help account for some of the Android device's processing time between creating files. The `createVideoFile()` method is quite similar to this one, except it creates a file with an 'mp4' video extension and obviously does not associate any body position to its filename.

Some methods were not described as they were not deemed relevant enough to be detailed or their implementation was based on extracted examples, such as the `getCameraInstance()` or `releaseMediaRecorder()` methods which were taken from the Android's Camera API guide.

[10]This method returns the current time in milliseconds since January 1, 1970 00:00:00.0 UTC. More at https://developer.android.com/reference/java/lang/System.html#currentTimeMillis()

5.3.2.2 BluetoothCommService class

The `BluetoothCommService` class is responsible for creating the objects that effectively handle the Bluetooth connections with the MoMos. It is largely based on the `BluetoothChatService` class from the Bluetooth Chat example, which was modified with the necessary steps to accommodate the MoMos' specifications.

Initially, the class's constructor was modified, as seen in Listing 5.12

```
//Constructor; sets up a new sensor. Added a String as an argument to identify each body part)
public BluetoothCommService(Context context, Handler handler, int bodyPositionCode, int operationMode) {
    mAdapter = BluetoothAdapter.getDefaultAdapter();
    mState = STATE_NONE;
    mHandler = handler;
    this.bodyPositionCode = bodyPositionCode; // Relates the connected MoMo to the body position through the requestCode
    this.operationMode = operationMode;  // In preparation for the operation mode
                                         //  choosing string to be sent.
}
```

Listing 5.12: `BluetoothCommService` constructor

Two attributes were added to the constructor in order to enable each instance of `BluetoothCommService` to carry through the selected body position and MoMo operation mode.

This class also implements three other subclasses in a similar manner to BluetoothChat example, namely `ConnectThread`, `AcceptThread` and `ConnectedThread`. These classes extend on Android's `Thread` class and they initiate threads for each of the Bluetooth connection phases - connecting, accepting and staying connected, respectively. There was one key modification that was made to the `ConnectedThread`. As discussed in Section 6.1.1, after observing a high error rate in the received packets a workaround[11] was implemented.

```
while (true) {
    try {
        if (mmInStream.available() > 0){
            try {
                // Read from the InputStream
                bytes = mmInStream.read(buffer);
                mHandler.obtainMessage(MainActivity.MESSAGE_READ, bytes, -1, buffer)
                        .sendToTarget();
            } catch (IOException e) {
                Log.e(TAG, "disconnected", e);
                connectionLost();
                // Start the service over to restart listening mode
                BluetoothCommService.this.start();
                break;
```

[11] Suggested by a Stack Overflow user at http://stackoverflow.com/q/12294705/3066185

```
15              }
16          }
17          else {
18              //if(DEBUG) Log.e(TAG, "sleeping for 200ms" + this);
19              SystemClock.sleep(200);
20          }
21      } catch (IOException e) {
22          e.printStackTrace();
23      }
24  }
```

Listing 5.13: `ConnectedThread while` cycle

Listing 5.13 shows the `while` cycle that is continuously run during the execution of the `ConnectedThread` instance. It reads the `InputStream`[12] buffer and sends it back to the connection's handler in `MainActivity`.

As a relatively high error rate was found specially when video recording was enabled, it was concluded that the Android device did not process (i.e., write it into its corresponding text file) the buffer quickly enough. As the cycle was continuously running and checking for new data on the `InputStream`, it overwrote the data buffer the handler was receiving.

The `sleep(200)` method suspends the thread for 200 ms. Hence, in one cycle, the app reads the `InputStream` data and sends it to the handler. On the next cycle, as it is very unlikely that new data is in the buffer (MoMos stream at a rate of 42 Hz), the cycle executes the `sleep` method and holds the thread there. This period allows for the device to save the received buffer into a file and prevents the cycle from reading the buffer before that period is over. The `InputStream` buffer builds up during the 200 ms and is sent to the handler all at once.

5.4 Conclusion

The developed application is thought to successfully meet its proposed requirements. The development of the app presented with several challenges - as mentioned, the use of simultaneous Bluetooth data streams was not found to be a common usage of this communication protocol.

In sum, the application is considered to meet its primary objectives of recording gait data from the MoMos and from device's video camera. It features a very simple usage and connects easily with the MoMos, enabling clinicians and patients to use it, both in clinical and ambulatory settings.

As it was developed on the Android platform, it is capable of functioning in a very wide array of devices, contributing to keeping this system's cost very low when compared to other gait analysis systems.

At this point, the app's operation was not yet assessed - a thorough analysis on its performance was carried out and is presented in Chapter 6.

[12] More about the `InputStream` class at https://developer.android.com/reference/java/io/InputStream.html

Chapter 6

System Analysis

6.1 Application Performance

In order to evaluate the performance of the developed Android application, several tests were performed. In these sections, these tests are described and their results are presented.

6.1.1 Data Reception Reliability

The first aspect that was accessed within the application was the reliability of the communication between the MoMos and the Android device.

For that purposed, four MoMos were connected to the Android device using the NeuroGait app. They were placed at a distance of 3 meters and were simultaneously recorded for 300 seconds. One of the MoMo's data stream was manually analyzed for errors - these were easily detectable as typically a bad packet presented a clear visual disruption. The errors for each thousand packets were also counted in order to observe the variability throughout the test.

The test yielded 12746 packets, 231 of which failed, resulting in a packet failure rate of 1.8%. Figure 6.1 shows the number of failed packets per 1000 received. There is some variation through-

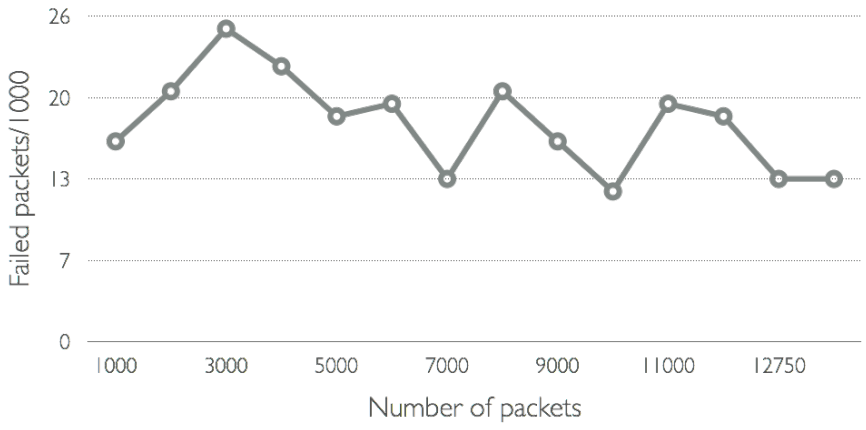


Figure 6.1: Plot of the y-axis accelerometer of the shank's MoMo

out the test but it was not deemed very significant.

Another aspect that was observed was that often several packet failures occurred consecutively, usually 2 to 4 at a time.

These results were not very satisfactory and therefore a solution was warranted. Eventually this led to implement the workaround described in Section 5.3.2.1. The error rate was drastically reduced to 1 to 2 failed packets per 1000, even with four simultaneous MoMos and video recording.

6.1.2 Temporal Analysis of Recordings

Several recordings were made using the application in order to evaluate other aspects of its performance.

Similarly to the tests described in Section 4.4, the MoMos were placed on a table. The table was knocked one time, a few seconds after the record button was pushed. By analyzing the data, the number of packets between the first one recorded and the packet which occurred when the table was knocked were determined. Then, to obtain the time interval between the first packet and the knock, the determined number of packets was multiplied by the values of frequency calculated in Section 4.4.

Video was also taped using the GoPro camera and analyzed using Avidemux.

6.1.2.1 Interval between video and data recording

	Q1	Q2	Q5	Q6
Test 1 (s)	0,79	-	0,91	0,95
Test 2 (s)	0,92	0,83	0,91	0,82

Table 6.1: Calculated time between start of MoMos and video recording (in seconds)

In order to assess if there were delays between the start of the recording of the MoMos' data streams and the start of the video recording, two tests were performed. A short recording with video and the data streams of four MoMos using the app was made. Using Avidemux, the time between the start of the video and the first visible knock was measured. Then, the number of packets between the first recorded packet and the first knock was determined, and the time between those events was calculated using the known frequencies for each MoMo.

After, the differences between these two sets of values were measured, to determine if there were significant differences between both processes.

Table 6.1 shows the results of these tests. For both, there is a clear delay, oscillating between 0,8 and 0,9 seconds.

6.1.2.2 Recording delay

Another aspect that was tested was how long it took between pressing the Record button within the app and the data recording actually starting.

Test (n)	1	2	3	4	5
Knock (s)	9,52	8,23	10,18	7,95	8,58
GP_y (s)	4,72	3,93	6,63	6,43	5,66
GP_r (s)	4,28	3,50	6,16	6,00	5,23

Table 6.2: GoPro video measurements for the recording delay tests

Five tests were performed according to what was described in the beginning of this Section. The GoPro camera was also used. In the videos, it was observed that when the button was pressed it would first turn yellow, immediately after it was pressed, and then changed to red. Bearing this in mind, it was first calculated the MoMo sample period (M_{sp}), which was the time between the start of each MoMo data stream recording and the knock. In order to do this, for each MoMo the number of packets from the start of the data stream to the first knock was counted and divided by the obtained frequency for that MoMo.

Using the GoPro video, the time period between the Record button changing colors (to yellow and then red) and to the knock was measured.

After, these values were compared to the calculated times for the MoMos data streams, by subtracting each MoMo sample period to the times measured by the GoPro.

6.2 Results and Discussion

Table 6.2 shows the measurements obtained from the GoPro camera and Table 6.3 shows the results for each MoMo. GP_y and GP_r are the GoPro measured times between the knock (the time in the video where the knock on the table occurred) and the appearance of the yellow button and the red button, respectively. Table 6.3 evidences the differences between these values. A $GP - M_{sp}$ value being positive indicates that the event occurred before the app started recording the MoMos data stream, while a negative signals that it happened after. The mean, maximum, minimum and standard deviation calculations between tests for each MoMo are also presented.

From these results, it can be concluded that the device starts recording the MoMos' data stream shortly after the Record button is pressed and before the Record button turns red.

Table 6.4 shows the same statistical analysis as in the previous tables but for each test (between MoMos). An important conclusion that can be drawn from these results, specially for the standard deviation values observed in Table 6.4 is that the device does not always start recording the MoMos simultaneously. This can be in partly explained by the MoMos sampling frequency, which admitting it is around 42 Hz, could result in a maximum discrepancy of two MoMo measurements that the Android device fails to register, or $\frac{2}{f} \simeq 48\ ms$. While this value is larger than the calculated standard deviations, it does not account for all the discrepancies between MoMos, which indicates that the processes within the execution of the app are causing the streams to be recorded asynchronously.

The application records the MoMos data stream reliably, without any significant packet losses. However, delays between the time the button is pressed and the time the data are actually recorded

Q1 Test (n)	1	2	3	4	5	Mean	Max (abs)	Min (abs)	StDev
M_{sp} (s)	4,66	3,93	6,47	6,26	5,65				
$GP_y - M_{sp}$ (s)	0,06	0,00	0,16	0,17	0,02	0,08	0,17	0,00	0,08
$GP_r - M_{sp}$ (s)	-0,38	-0,43	-0,31	-0,26	-0,42	-0,36	0,43	0,26	0,07
Q2 Test (n)	1	2	3	4	5	Mean	Max (abs)	Min (abs)	StDev
M_{sp} (s)	4,58	3,87	6,40	6,33	5,57				
$GP_y - M_{sp}$ (s)	0,14	0,06	0,23	0,10	0,09	0,12	0,23	0,06	0,07
$GP_r - M_{sp}$ (s)	-0,30	-0,37	-0,24	-0,33	-0,34	-0,32	0,37	0,24	0,05
Q5 Test (n)	1	2	3	4	5	Mean	Max (abs)	Min (abs)	StDev
M_{sp} (s)	4,66	3,83	6,44	6,26	5,60				
$GP_y - M_{sp}$ (s)	0,06	0,10	0,19	0,17	0,06	0,12	0,19	0,06	0,06
$GP_r - M_{sp}$ (s)	-0,38	-0,33	-0,28	-0,26	-0,37	-0,32	0,38	0,26	0,05
Q6 Test (n)	1	2	3	4	5	Mean	Max (abs)	Min (abs)	StDev
M_{sp} (s)	4,58	3,86	6,42	6,35	5,68				
$GP_y - M_{sp}$ (s)	0,14	0,07	0,22	0,08	-0,02	0,10	0,22	0,02	0,09
$GP_r - M_{sp}$ (s)	-0,30	-0,36	-0,26	-0,35	-0,45	-0,34	0,45	0,26	0,07

Table 6.3: Recording delay results and analysis between tests for each MoMo

$GP_y - M_{sp}$	1	2	3	4	5
Mean (s)	**0,102**	**0,057**	**0,198**	**0,132**	**0,036**
Max (s)	0,14	0,10	**0,23**	0,17	0,09
Min (s)	0,06	**0,00**	0,16	0,08	-0,02
StDev (s)	0,04	0,03	0,026	0,042	0,041
$GP_r - M_{sp}$					
Mean (s)	**-0,338**	**-0,373**	**-0,272**	**-0,298**	**-0,394**
Max (s)	0,38	0,43	0,31	0,35	**0,45**
Min (s)	0,30	0,33	**0,24**	0,26	0,34
StDev (s)	0,039	0,034	0,026	0,042	0,041

Table 6.4: Statistical analysis for each test (in seconds)

have been detected. Moreover, it was found the data recordings are not initiated at precisely the same time, which leads to a significant asynchronism between the data streams.

Chapter 7

Gait Analysis

An analysis on normal human gait using the MoMos is presented in this chapter. A trial consisting of a few steps along a line back and forth was made using two MoMos placed on the right leg and shank of a healthy male adult. The data extracted were analyzed using Matlab. Its purpose was to determine if the MoMos were able to identify and extract gait parameters such as pinpointing specific phases of the gait cycle.

7.1 Methodology

Two MoMos were placed on the subject's right leg, facing rightwards - one on the shank and another on the thigh. The accelerometer y-axis pointed upwards, the z-axis pointed rightwards and the x-axis forward (see Figure 4.4), At this stage, the mobile app was still not developed and the data was captured using a serial port terminal application, named CoolTerm[1].

Prior to the test, a connection to the MoMos was established. A few seconds before the test initiated, the data streams were started being saved to a text file. As this was done manually for each MoMo, their data were recorded asynchronously.

Prior to walking, the subject nudged each MoMo so it would cause a disturbance that could be easily visualized in the data plots. The test was also video recorded, which enabled a comparison between the MoMos' data and the film recorded. The video and data stream recordings were finished right after the subject stopped walking.

7.2 Data Analysis

The data from the recordings were imported into Matlab and plotted.

Figure 7.1 shows the plot of y-axis accelerometer of the shank's MoMo of the complete trial. The MoMos output values were converted to g, in order to better perceive the range of values. The horizontal axis, named frame number, relates to the number of packets - or frames - since the trial had started.

[1] More information on http://freeware.the-meiers.org/

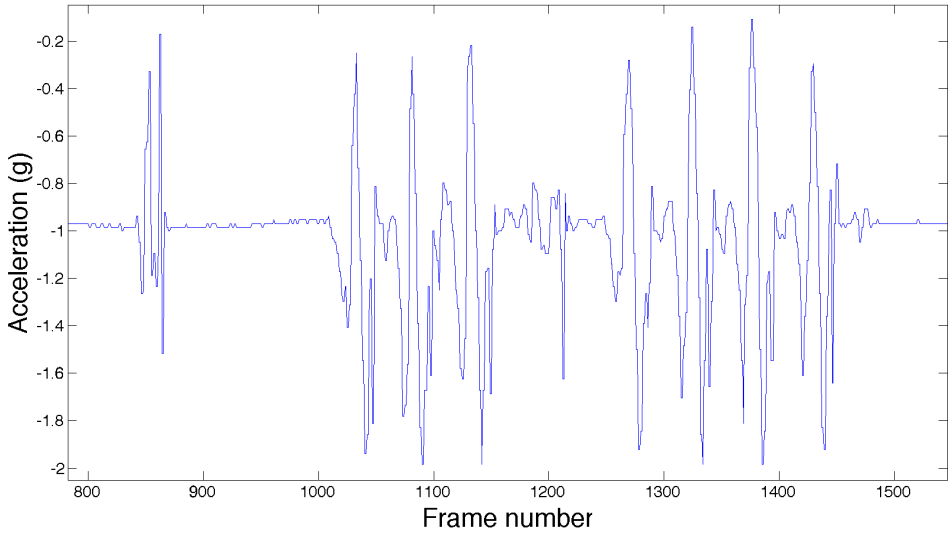


Figure 7.1: Plot of the y-axis accelerometer of the shank's MoMo

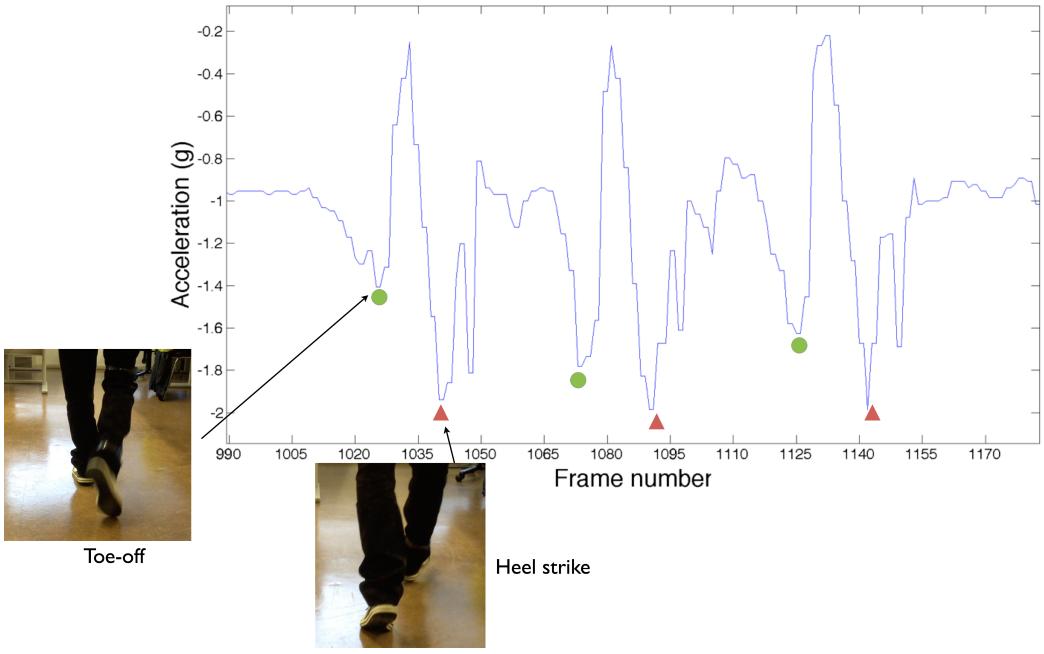


Figure 7.2: Data plot of first three steps of the right leg shank's MoMo accelerometer y-axis

The first noticeable peaks are due to the nudge given to the MoMo prior to the start of walking. It can also be observed that when there is no variation throughout the MoMo output axis, only the effect of gravity is observable as the value hovers around -1 g which is its expected value on the y-axis. Between sensibly frame 1020 and frame 1150, the first three steps are visible. The subject then turns around until about frame 1250 and takes four more steps in the opposite direction.

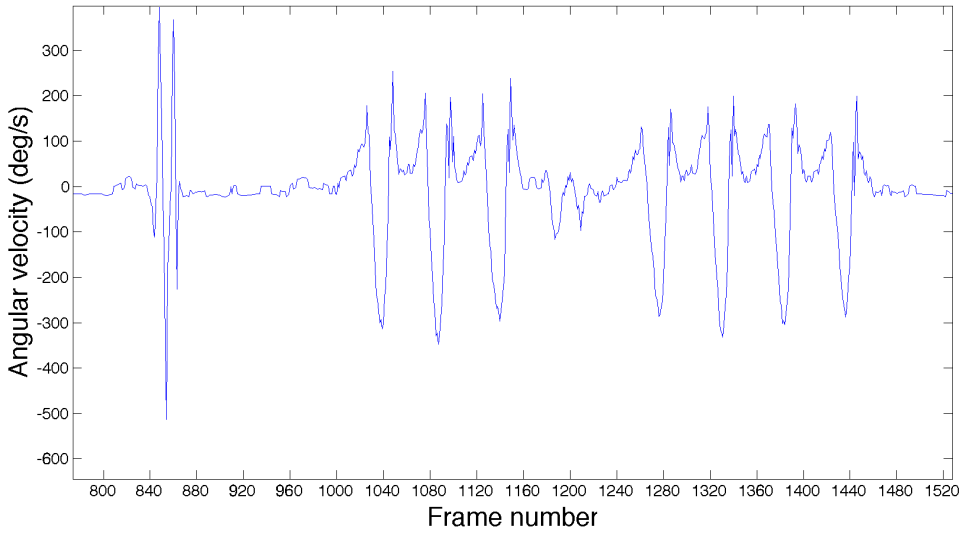


Figure 7.3: Plot of z-axis gyroscope of the shank's MoMo

Figure 7.2 zooms in on the previous plot and evidences the first three steps of the trial. With the aid of the recorded video, the right foot's toe-off and heel strike events were identified and marked in the plot. The green circles show toe-off events and the red triangles heel strikes. The first two events are illustrated with the frames from the recorded video which corresponded to those events. These results are coherent with the ones reported in [13] and shown in Figure 2.5.

Next, the gyroscope plot for the z-axis of the shank MoMo is analyzed. This axis is parallel to the ground and points rightwards, relatively to the body. Although the z-axis gyroscope plot shown in Figure 7.3 shares some similarities with Figure 7.1 - such as the time the nudge to the MoMo occurred and the start and stop of the gait, some features of the gait cycle are much more evident on the gyroscope plot. Namely, the swing phase and the support phase are very clearly distinguishable from each other, as seen on Figure 7.4, where the first three steps are detailed and the swing and support phases of the first step are identified. The plot outline of these gait cycle phases is very similar to the ones obtained in [10, 13, 20], where gyroscopes were also placed in the subjects' shank, albeit the direction of the axis in [10] is reversed.

7.3 Conclusions

A simple trial using two MoMos on a subject's right leg was conducted. An analysis on the obtained data has shown that the MoMos are capable of evidencing important gait features, specif-

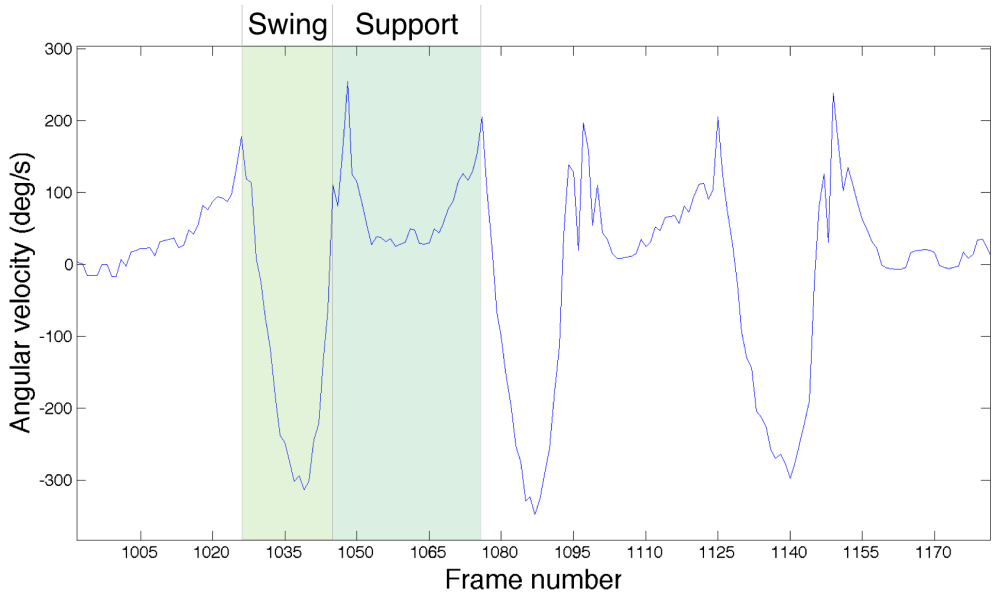


Figure 7.4: Plot of z-axis gyroscope of the shank's MoMo

ically of identifying heel strike and toe-off events and the swing and support phases of the gait cycle. Some results closely related to others reviewed in Chapter 2.

At this point, there were no concerns with extracting other gait parameters, such as stride length, speed and frequency. Such parameters are discussed in Chapter 8.

Chapter 8

Experimental Trials

The NeuroGait system was presented to this thesis co-supervisor, Prof. Dr. Kai Bötzel on a week-long visit to Munich, at the University Clinic Großhadern in Ludwig-Maximilians-University (LMU).

Initially, only the NeuroGait system presentation was scheduled. However, after some discussion with this thesis supervisor and co-supervisor, it was decided to perform a set of trials with the NeuroGait system, together with Prof. Dr. Bötzel's GaitWatch system, in LMU's research facilities.

This proved to be an invaluable opportunity as the access to high precision equipment, which could accurately assess the NeuroGait system in a controlled clinical environment, was provided; along with Prof. Dr. Bötzel's expertise, who is a leading specialist on the neurophisiology of movement and PD.

Prof. Dr. Bötzel's defined the experimental protocol and coordinated these trials. These experiments took place throughout the following four days.

These trials were performed in order to evaluate the developed system using a high performance motion capture camera system as ground truth.

Figure 8.1 provides an overview of the laboratory where the tests took place.

8.1 Objectives

The main goal of this study was to validate NeuroGait system as a gait assessment tool and to assess its performance against an optical motion capture system, which provides highly accurate results.

Key parameters of gait were evaluated - swing length and speed were extracted from the data. The values from NeuroGait and from the Oqus camera system were computed and compared to each other.

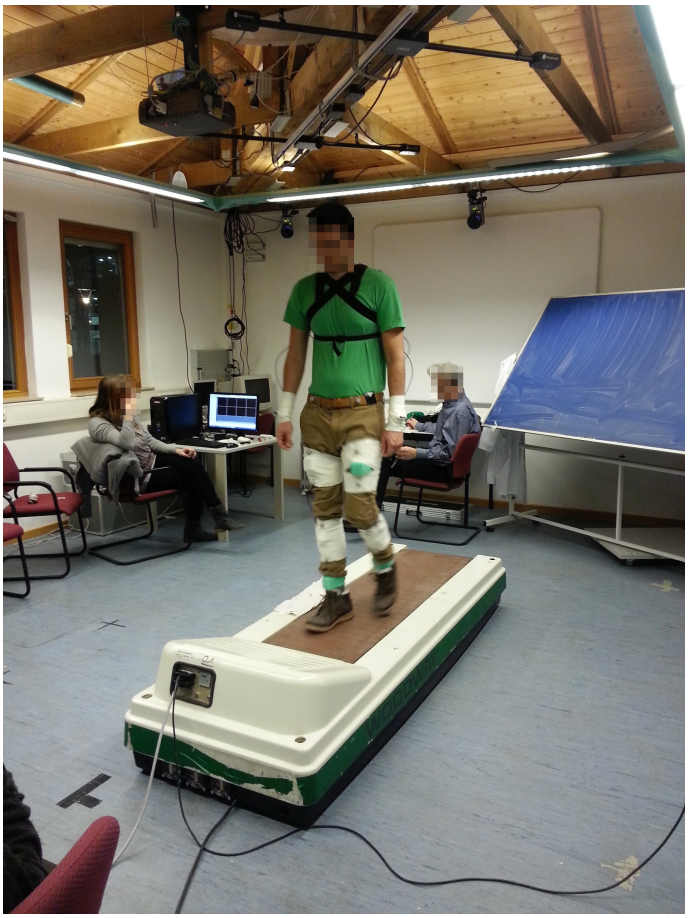


Figure 8.1: Overview of LMU's laboratory equipped with an Oqus camera system

8.2 Methodology

In this section, the methods and processes used to gather data, the study population and tools utilized are described.

8.2.1 Materials

Several instruments were used in these experiments. Next, the main materials are described.

8.2.1.1 Oqus Motion Capture System

Qualysis produces high-end motion capture (mocap) systems, which are, among other purposes, used for gait analysis[1].

In these trials, Oqus cameras were used to capture the subjects' gait, outputting three dimensional marker coordinates for each test. The markers used were passive reflective spheres[2].

[1] More information at http://www.qualisys.com/applications/biomechanics/gait-analysis-and-rehabilitation/

[2] Available from Qualisys (http://www.qualisys.com/products/accessories/passive-markers/lightweight/)

Figure 8.2: Oqus Camera[3]

The room where the experiments were conducted was equipped with a set of eight Oqus Cameras linked to a computer with Qualisys Track Manager installed. Some of these can be seen on Figure 8.1, between the ceiling and the furthest wall.

8.2.1.2 Treadmill

A Woodway S1V treadmill was used to perform these tests. This treadmill had an external controller, which was used to visualize and regulate the speed. Its handles were detached from the body, in order to prevent reflection and occlusion to the Oqus cameras.

8.2.1.3 NeuroGait System

The four MoMos that were used before were also used in these trials. The NeuroGait app was installed in a Samsung Galaxy Note-II with Android 4.3 installed.

The MoMos were fitted on the subjects while inside wrist bands, which were wrapped around a set of elastic bands with hook-and-eye clasps, as seen in Figure 8.3a.

8.2.2 Population

Ten adult healthy males took part in this study. Table 8.1 details the subjects' height and leg measurements.

8.2.3 Data collection

In the following sections, the methods and steps for equipping the subjects and gathering data are described.

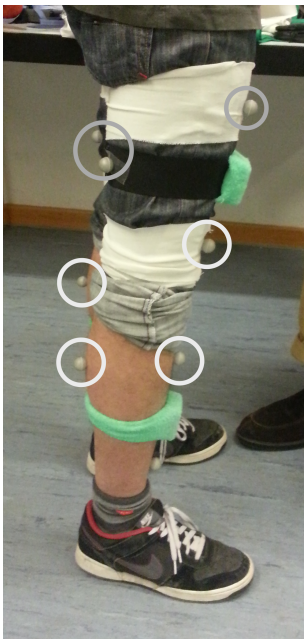
[3]Image obtained from http://www.thoma.de/en/produkte/walkfinder_camera6.html

Subject	Height (cm)	Total Leg Length (cm)	Lower Leg Length (cm)
1	193	104	-
2	183	95	55
3	182	93	57
4	173	89	51
5	183	92	54
6	190	94	53
7	196	106	62
8	178	88	51
9	185	88	58
10	187	98	58
Average	185	94,7	55,4
St Dev	6,9	6,3	13,0
Max	196	106	95
Min	173	88	51

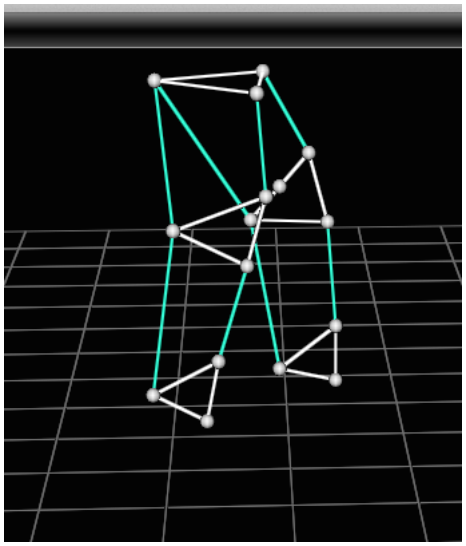
Table 8.1: Subject measurements

8.2.3.1 Materials Placement

Each subject was fitted with four MoMos - two on each leg, one in the front side of the thigh and another in the front side of the shank. MoMos were placed facing forwards, with the accelerometer y-axis pointing up and the z-axis pointing forwards.



(a) Placement of reflective markers and MoMos on a subject's legs



(b) 3d plot of reflective markers

Figure 8.3: Placement of MoMos and 3d representation generated by Mokka

Subjects were also equipped with reflective markers to enable three dimensional motion capture by the Oqus camera system. For this purpose, the lower body was divided into five sections - hip, right and left thigh and right and left shank. In each section, three reflective markers were placed - two in the front and one in the back, effectively forming a triangle. Figure 8.3 shows these markers on the subjects legs (Figure 8.3a) and their corresponding three dimensional plot (Figure 8.3b). While the hip markers were placed horizontally in relation to the ground, the remaining markers formed vertical shapes.

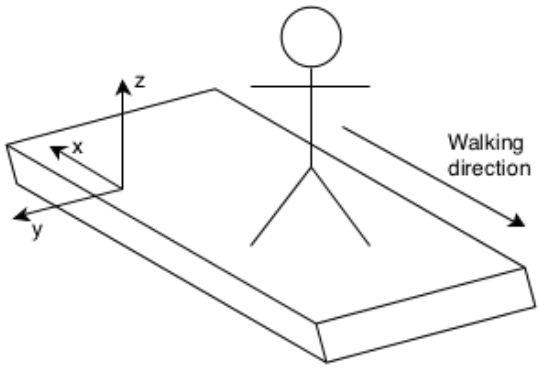


Figure 8.4: Representation of the Oqus coordinate axes on the treadmill

The Oqus camera system was calibrated using the surface of the treadmill as the base for the xy plane. The subjects would walk on the opposite direction of the x-axis, as shown on Figure 8.4.

8.2.3.2 Procedures

Each subject was asked to walk on a treadmill three times, each at different fixed speeds - 2 km/h, 4 km/h and 6 km/h, for a period of about a minute.

The trials started at a null speed. Before subjects started walking, they were asked to stomp on the treadmill, one foot at a time, to imprint a distinguishable marker in the collected data. The treadmill was then slowly accelerated to a fixed speed and at that point the subject walked for a period of 40 to 50 seconds. Afterwards the treadmill would slowly be decelerated back to zero speed. The NeuroGait and the Qualisys systems captured data for roughly the same period.

The NeuroGait app collected the data stream from the four MoMos and also recorded video. The Qualisys system gathered the markers' coordinates, set at a frequency of 200 Hz.

The treadmill controller was used to set and record the speed throughout the trials.

Figure 8.1 depicts one of the trials while it was in progress - the subject is walking on the treadmill while fully equipped with passive markers, the MoMos and the GaitWatch system.

8.3 Data Analysis

The goal of the analysis on this experiment's data was to extract important gait parameters and compare the NeuroGait system to an accurate ground truth system.

First, the data generated by the mocap system was analysed.

8.3.1 Mocap Data

The Qualysis Track Manager outputted data in C3D file format[4].

Mokka, or Motion kinematic and kinetic analyzer[5], is an open-source application capable of analyzing C3D data files.

The software displays each marker in a three-dimensional space and allows spatial data from each marker to be extracted. It can also animate those markers in a three-dimensional view. Also, it has the ability to mark gait events, such as heel strike and toe-off for each foot. Another feature is the insertion of video side-by-side within the application window. It also allows the synchronization between the video and the three-dimensional markers playback by adjusting a time value for the delay the video should have.

8.3.1.1 Marker Coordinates Analysis

In order to determine how gait parameters could be extracted from C3D data, one of the 4 km/h tests was loaded into Mokka.

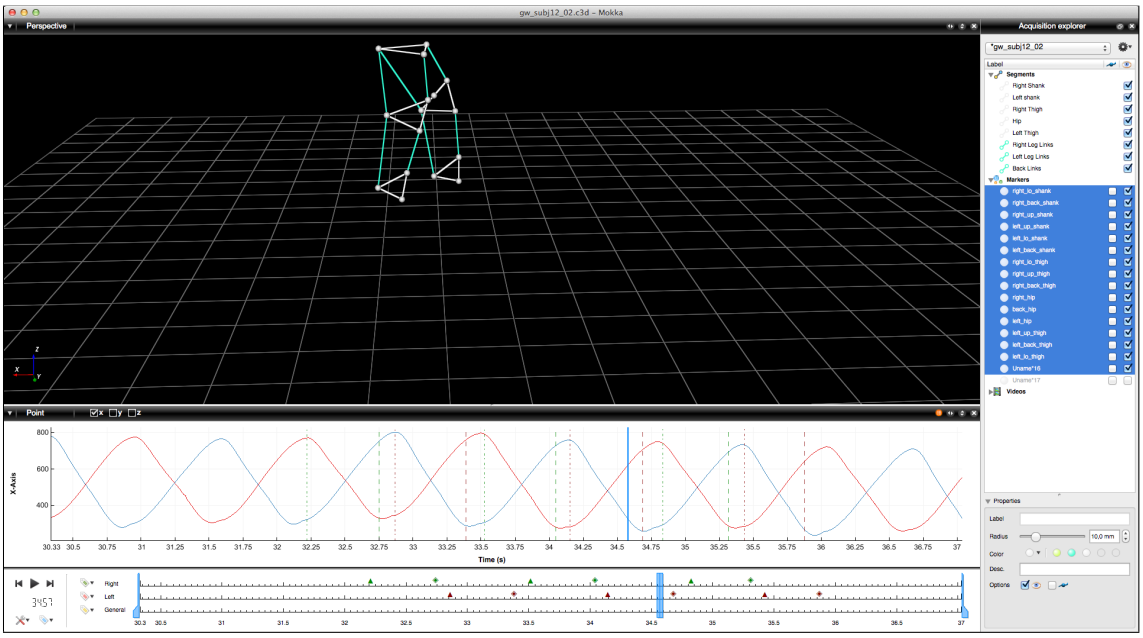


Figure 8.5: Screenshot of Mokka with three-dimensional data loaded and segments traced

[4]C3D is a public domain file format for 3D data. More information at: http://www.c3d.org

[5]More at http://b-tk.googlecode.com/svn/web/mokka/index.html

Firstly, the recorded video was synchronized to the 3D markers playback, using the right foot stomp performed by the subject, that was clearly visible in both the video recording and the markers playback. Both were shown side-by-side and the video delay was then adjusted according to the difference in time between the video and the markers playback of that event.

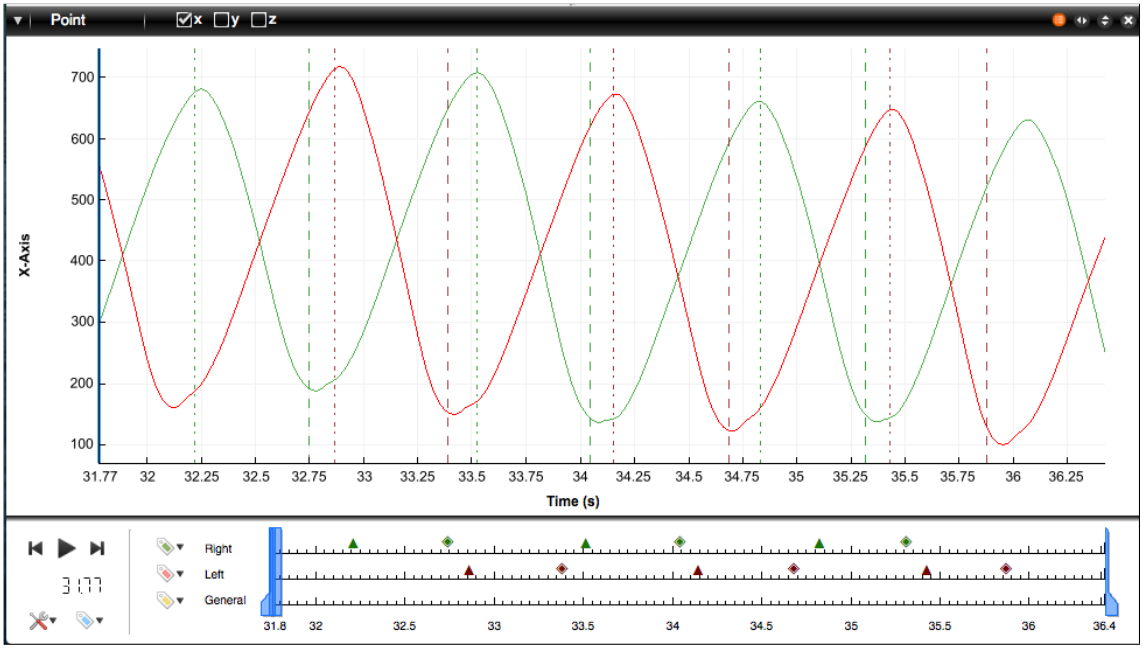


Figure 8.6: Plot of the right and left foot lower shank markers in the x-axis

After, a section of a few steps, where the speed had stabilized at 4 km/h, was chosen and zoomed into in the time bar. Using the synchronized video recording, the heel strike and toe-off events for three gait cycles for each foot were pinpointed. This can be observed on the bottom part of Figure 8.6. The identification of these events was aided and validated by a physical therapist.

Next, the marked events were compared to the x-axis plot of the right and left lower shank markers. Mokka marks heel strike events as squares on the time bar and vertical dashed lines on the graph plot; it also marks toe-off events as triangles on the time bar and dashed and dotted lines on the graph plot. The right foot events and plot are drawn in green, while the left foot are in red.

At this point, an important observation could be drawn from the plot: nearly all of the toe-off and heel strike events occur at each local maximum and local minimum, respectively. From this, it was concluded that these events could be extracted by analyzing the local extrema of the lower shank markers plots.

8.3.1.2 Marker coordinates feature extraction

Using Mokka, the C3D data from the right and left lower shank markers were exported into a text file. The data were then imported to Matlab.

Using the PeakFinder function[6], the local extrema for the x-axis data were plotted. PeakFinder is able to extract the values of the local extrema and their position (or frame number).

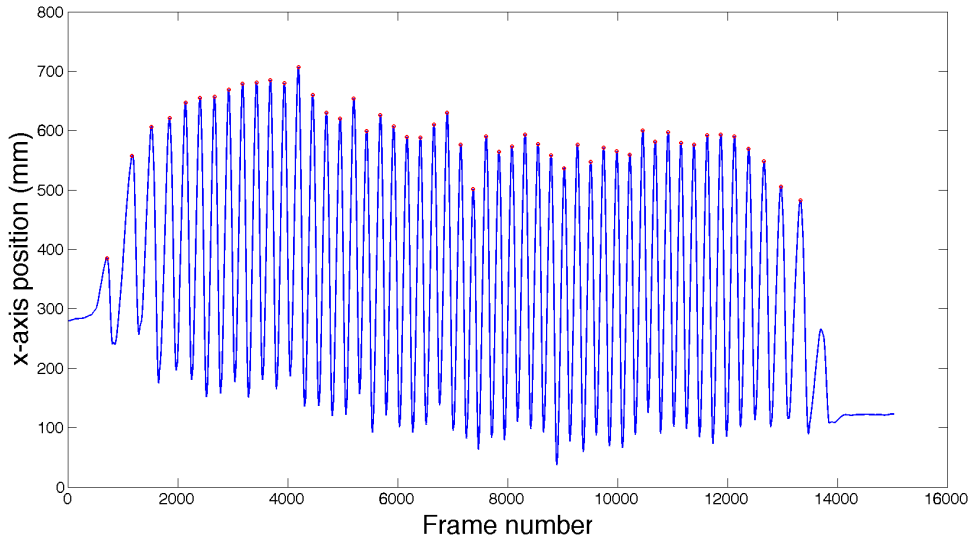


Figure 8.7: Plot of the right foot lower shank marker in the x-axis with local maxima outlined

Figure 8.7 shows the PeakFinder output plot with the local maxima, i.e., the toe-off events marked with red dots. The plot is limited to when the subject was walking.

After extracting the local extrema and its corresponding positions for both the left and right shank plots, these data were pasted into Microsoft Excel. The positions were converted into time by dividing them by the mocap sampling frequency (200 Hz) The extrema pertaining to the 20th to 29th steps were selected, as the treadmill's speed during those step was known to be stabilized at 4 km/h.

From there several parameters were calculated, namely the swing length, swing time and swing velocity. The swing length was calculated as the modulus of the difference between the heel strike and the toe-off position for a given step; the swing time was the difference between the heel strike and the toe-off time for a given step; and the swing velocity as the quotient between swing length and swing time. The results are shown in Table 8.2, which also includes the mean, maximum, minimum and standard deviation calculations, for each measurement and result.

8.3.2 MoMos Data

The data from the gyroscope of right shank MoMo from the same trial were imported into Matlab and ran through PeakFinder. Unlike the C3D data, where the local maxima and minima relate to toe-off and heel strike events, in the gyroscope case, only the local maxima obtained at the beginning and the end of the swing phase respectively indicate toe-off and heel strike events.

[6]More on this function at: http://www.mathworks.com/matlabcentral/fileexchange/25500-peakfinder

Step	Toe-off (mm)	Heel strike (mm)	Swing Length (mm)	Toe-off Time (s)	Heel-strike Time (s)	Swing Time (s)	Swing Velocity (m/s)
1	607,41	101,19	506,22	29,64	30,17	**0,53**	**0,96**
2	589,56	92,29	497,27	30,86	31,38	**0,52**	**0,96**
3	588,49	105,44	483,05	32,07	32,59	**0,52**	**0,93**
4	610,34	137,98	472,36	33,31	33,82	**0,52**	**0,92**
5	629,91	94,86	535,05	34,52	35,05	**0,54**	**1,00**
6	576,14	82,18	493,96	35,75	36,25	**0,51**	**0,98**
7	501,94	63,58	438,34	36,88	37,35	**0,48**	**0,92**
8	590,13	83,66	506,47	38,05	38,55	**0,50**	**1,01**
9	564,65	79,37	485,28	39,22	39,72	**0,50**	**0,97**
10	573,68	109,59	464,09	40,40	40,91	**0,51**	**0,92**
Mean	583,23	95,014	488,211	35,067	35,576	**0,511**	**0,956**
Max	629,91	137,98	535,05	40,40	40,91	**0,54**	**1,01**
Min	501,94	63,58	438,36	29,64	30,17	**0,48**	**0,92**
StDev	34,49	20,38	26,50	3,612	3,607	**0,017**	**0,034**

Table 8.2: Results of right lower shank C3D data

The input parameters of PeakFinder were iteratively adjusted and ultimately set to `sel` = 500 and `threshold` = 600 which resulted in the plot of Figure 8.8, which features the right leg stomp and the first step. They were set to `sel` = 500 and `threshold` = 600.

It can be noticed that it misses the first swing local maxima as the swing velocity was too slow to generate peaks within the selected PeakFinder thresholds and it incorrectly registers (i.e., it is not a gait event) the first maxima, which corresponds to the stomp prior to the start of the gait. This was taken into account when determining the steps to be analyzed.

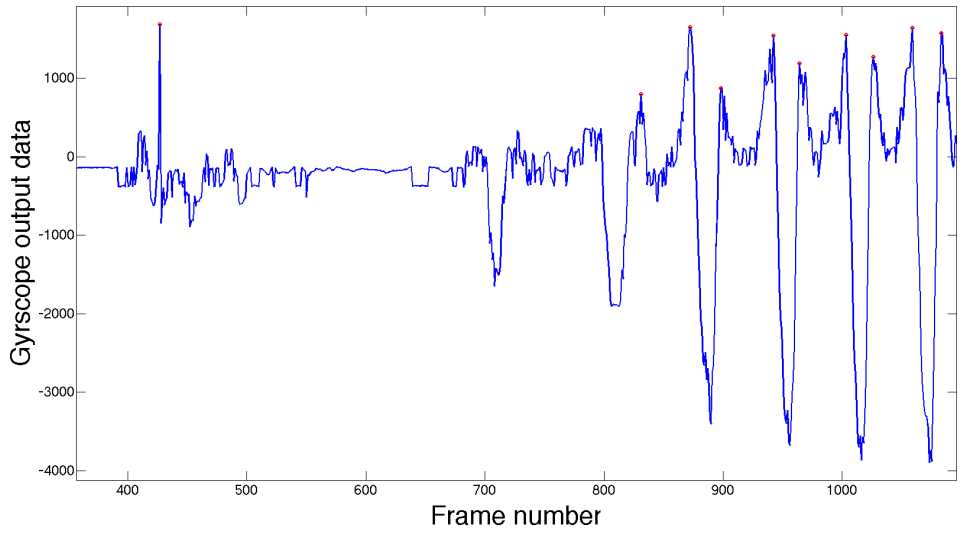


Figure 8.8: Right shank MoMo gyroscope y-axis data plot with local maxima outlined of right foot stomp and first steps

Similarly to Section 8.3.1, the PeakFinder output data was imported into Excel and analyzed. In order to select the 20th to 29th toe-off and heel strike events from the local maxima, firstly, the toe-off corresponding to start of the 20th gait cycle was found. The next maximum would correspond to the right foot's heel strike event of the 20th gait cycle. The next 9 steps were then recursively selected.

Step	Number of frames	Swing Time (s)	Extrapolated Distance (m)	Treadmill speed distance (m)
1	23	0,54	0,52	0,60
2	23	0,54	0,52	0,60
3	20	0,47	0,44	0,52
4	22	0,52	0,48	0,58
5	25	0,59	0,59	0,65
6	22	0,52	0,51	0,58
7	20	0,47	0,43	0,52
8	21	0,49	0,50	0,55
9	22	0,52	0,50	0,58
10	21	0,49	0,45	0,54
Mean	21,9	0,52	0,493	0,572
Maximum	25	0,59	0,59	0,65
Minimum	20	0,47	0,43	0,52
St dev	1,52	0,036	0,046	0,040

Table 8.3: Results of the gyroscope data of the right shank MoMo

From there, the swing time for each step was calculated, using the known value of frequency for the MoMo with the Q5 id - 42,51 Hz. The swing length was then extrapolated multiplying the obtained time with the C3D calculated velocities shown in Table 8.2.

This parameter was also estimated using the treadmill speed even though it is realized the speed at which the treadmill was rolling will likely differ from the actual speed at which the subject swings his leg.

8.4 Results and Discussion

Based on the the obtained results, comparisons between the data obtained from the Qualisys and the NeuroGait systems were performed.

Step	1	2	3	4	5	6	7	8	9	10
Relative Error (%)	2,08	4,05	9,52	0,49	9,92	2,48	0,95	1,20	3,51	2,18
			Max	Min	Mean	StDev				
			9,92	0,49	3,64	3,39	(%)			

Table 8.4: MoMo swing time relative error vs. ground truth

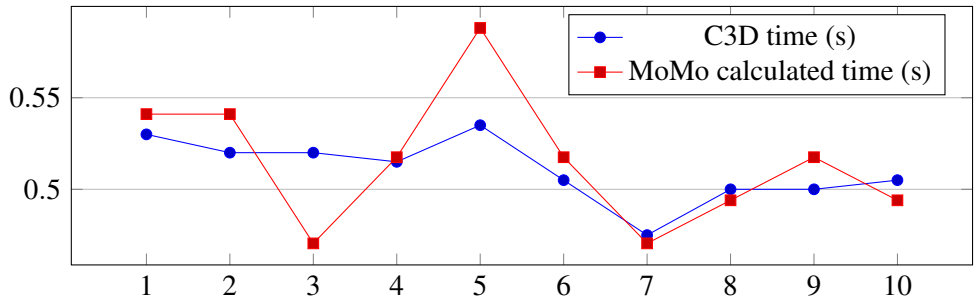


Figure 8.9: Swing times comparison (horizontal axis in steps and vertical axis in seconds)

Firstly, the time for each swing was compared. Figure 8.9 shows the time in seconds (on the vertical axis) for each swing. The plot reveals that the times calculated by the MoMo's data follow closely most of the values found in the Qualisys data.

Bearing in mind that the MoMo samples have a resolution of $1/f = 24ms$, it can be observed that the error for all the steps except the third and fifth steps fall under the sampling period of the MoMos. The time differences in steps 3 and 5 also fall under the period of two MoMo samples and seem to offset each other, as the differences between the ground truth time and the MoMo calculated time in steps 3 and 5 are nearly symmetrical.

Table 8.4 shows the calculated relative error for each step for the MoMo data against the Qualisys data.

While the maximum error seems relatively high, the average relative error 3,64% seems to indicate that the NeuroGait system performs fairly well when assessing swing times.

These data suggest that, even though the MoMos provide accurate results when determining swing times, at the set frequency the period of two samples - 48 ms - take up almost 10% of the duration of a normal swing, which was calculated to be 0,51 s on average. Therefore, this particular application could benefit from an increase in the MoMo's sampling frequency, as it would narrow the window for the maximum relative error.

Next, the swing lengths were compared. Figure 8.10 shows a comparison between the Qualisys measured swing lengths and the lengths obtained by multiplying the MoMo swing times and the treadmill speed.

The error between both sets of values is evidently high. However, the relation between both

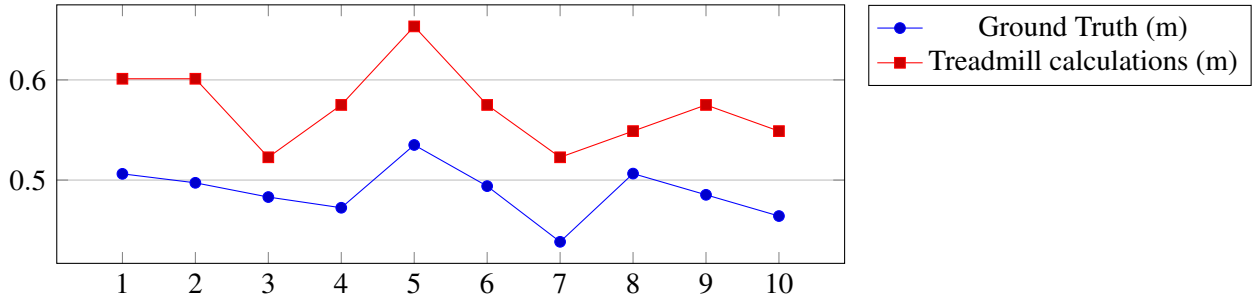


Figure 8.10: Swing lengths comparison (horizontal axis in steps and vertical axis in meters)

sets is such that it appears there is an offset for the MoMo swing lengths, suggesting a high systematic error, possibly caused by the indicated treadmill speed.

These experiments provided a large dataset of MoMo generated data and video.

An analysis on a small part of the data was performed, which resulted in some insights into the NeuroGait system's performance, although much more can be done in order to acquire a more accurate assessment of the NeuroGait system.

The manner of how the system was evaluated differs from the methods found in similar systems, and thus, as of yet, it cannot be directly compared to other systems. However, in one study the assessment done on the shank angular velocity yielded an RMS relative error of 5.0% at similar speeds [20].

A further analysis of the collected dataset is needed in order to effectively validate this system as a viable gait assessment tool. From the results obtained, some insights into key gait parameters such speed, cadence and stride length can already be extracted.

Chapter 9

Conclusions and Future Work

In this chapter, the conclusions of this work, namely the overview of the completed goals and an outline for suggested future work is presented.

9.1 Achievement of Goals

The main goals for this dissertation were generally attained. These included the development of a mobile application that gathered data from a set of MoMos, which was successfully done, although testing revealed that data synchronicity falls slightly under what would be desired.

The NeuroGait system has unique features, such as the fact that it supports HD video recording simultaneously with MoMo data recording.

The analysis performed in Chapter 7 provided a very detailed view of the system's capabilities and shortcomings, showing what needs to be improved.

The analysis of results obtained in Chapter 8 from the trials performed in Munich provides some insights into the system's performance with gait assessment; however this analysis is somewhat superficial and many more conclusions can be drawn from the collected data.

The accurate and complete assessment of the obtained data is a complex process which is not within the scope of this thesis.

9.2 Future Work

The mobile application performs well, as it reliably records the MoMos' data stream and HD video. Notwithstanding, one of the key goals was the synchronicity between all the data streams, which was not fully achieved as significant differences between the times when the streams start recording were found. As a result, in order to obtain a precise synchronism between data streams, the current usage of the system requires synchronization based on visual or kinematic cues, such as synchronizing using the recorded video or simply by stomping a foot.

This warrants a further analysis on the developed application to determine if it is possible to decrease these variations and improve the system's performance.

A considerable dataset was generated at LMU's laboratory, which provided a large pool of data, needed to validate this system. Methods for better evaluating the NeuroGait system and obtain superior gait feature extraction should be planned and devised.

The NeuroGait system could be better enabled as an ambulatory one with the implementation of real-time or near real-time data uploading and event detection within a cloud-based service, which would enable the clinician to retrieve immediate results, even if away from a patient utilizing the system.

The integration with other systems such as the one introduced in [21] could also provide a more complete and accurate assessment of a patient's gait, by combining inertial data and three-dimensional imaging.

Bibliography

[1] J. Parkinson, *An essay on the shaking palsy*. Printed by Whittingham and Rowland for Sherwood, Neely, and Jones, 1817.

[2] M. De Rijk, L. Launer, K. Berger, M. Breteler, J. Dartigues, M. Baldereschi, L. Fratiglioni, A. Lobo, J. Martinez-Lage, C. Trenkwalder *et al.*, "Prevalence of Parkinson's disease in Europe: A collaborative study of population-based cohorts. Neurologic Diseases in the Elderly Research Group." *Neurology*, vol. 54, no. 11 Suppl 5, pp. S21–3, 1999.

[3] B. S. Shastry, "Parkinson disease: etiology, pathogenesis and future of gene therapy," *Neuroscience research*, vol. 41, no. 1, pp. 5–12, 2001.

[4] R. Pahwa, K. E. Lyons, and W. Koller, *Handbook of Parkinson's disease*. CRC Press, 2012.

[5] J. M. Hausdorff, "Gait dynamics in Parkinson's disease: common and distinct behavior among stride length, gait variability, and fractal-like scaling," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 19, no. 2, p. 026113, 2009.

[6] S. von Campenhausen, Y. Winter, A. R. e Silva, C. Sampaio, E. Ruzicka, K. Bötzel, P. Barone, W. Poewe, A. Guekht, C. Mateus, K.-P. Pfeiffer *et al.*, "Costs of illness and care in Parkinson's Disease: An evaluation in six countries," *European Neuropsychopharmacology*, vol. 21, pp. 180–191, 2011.

[7] C. Ramaker, J. Marinus, A. M. Stiggelbout, and B. J. van Hilten, "Systematic evaluation of rating scales for impairment and disability in Parkinson's disease," *Movement Disorders*, vol. 17, no. 5, pp. 867–876, 2002.

[8] G. Geminiani, B. Cesana, F. Tamma, P. Contri, C. Pacchetti, F. Carella, R. Piolti, E. Martignoni, P. Giovannini, and F. Girotti, "Interobserver reliability between neurologists in training of parkinson's disease rating scales. a multicenter study." *Movement disorders: official journal of the Movement Disorder Society*, vol. 6, no. 4, pp. 330–335, 1990.

[9] J. Stamatakis, J. Cremers, D. Maquet, B. Macq, and G. Garraux, "Gait feature extraction in Parkinson's disease using low-cost accelerometers," in *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE*, Aug 2011, pp. 7900–7903.

[10] A. Salarian, H. Russmann, F. J. Vingerhoets, C. Dehollaini, Y. Blanc, P. R. Burkhard, and K. Aminian, "Gait assessment in Parkinson's disease: toward an ambulatory system for long-term monitoring," *Biomedical Engineering, IEEE Transactions on*, vol. 51, no. 8, pp. 1434–1443, 2004.

[11] S. Fahn, D. Oakes, I. Shoulson, K. Kieburtz, A. Rudolph, A. Lang, C. Olanow, C. Tanner, and K. Marek, "Levodopa and the progression of parkinson's disease." *The New England journal of medicine*, vol. 351, no. 24, pp. 2498–2508, 2004.

[12] M. Whittle, "Gait analysis: An introduction. 2007," *Heidi Harrison*, pp. 47–100.

[13] J. Rueterbories, E. G. Spaich, B. Larsen, and O. K. Andersen, "Methods for gait event detection and analysis in ambulatory systems," *Medical engineering & physics*, vol. 32, no. 6, pp. 545–552, 2010.

[14] W. Maetzler, J. Domingos, K. Srulijes, J. J. Ferreira, and B. R. Bloem, "Quantitative wearable sensors for objective assessment of Parkinson's disease," *Movement Disorders*, vol. 28, no. 12, pp. 1628–1637, 2013. [Online]. Available: http://dx.doi.org/10.1002/mds.25628

[15] P. Esser, H. Dawes, J. Collett, M. G. Feltham, and K. Howells, "Validity and inter-rater reliability of inertial gait measurements in Parkinson's disease: A pilot study "," *Journal of Neuroscience Methods*, vol. 205, no. 1, pp. 177 – 181, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165027012000076

[16] C.-C. Yang, Y.-L. Hsu, K.-S. Shih, and J.-M. Lu, "Real-time gait cycle parameter recognition using a wearable accelerometry system," *Sensors*, vol. 11, no. 8, pp. 7314–7326, 2011.

[17] K. J. O'Donovan, R. Kamnik, D. T. O'Keeffe, and G. M. Lyons, "An inertial and magnetic sensor based technique for joint angle measurement," *Journal of biomechanics*, vol. 40, no. 12, pp. 2604–2611, 2007.

[18] R. Williamson and B. J. Andrews, "Gait event detection for FES using accelerometers and supervised machine learning," *Rehabilitation Engineering, IEEE Transactions on*, vol. 8, no. 3, pp. 312–319, 2000.

[19] S.-W. Lee, K. Mase, and K. Kogure, "Detection of spatio-temporal gait parameters by using wearable motion sensors," in *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*. IEEE, 2006, pp. 6836–6839.

[20] R. E. Mayagoitia, A. V. Nene, and P. H. Veltink, "Accelerometer and rate gyroscope measurement of kinematics: an inexpensive alternative to optical motion analysis systems," *Journal of biomechanics*, vol. 35, no. 4, pp. 537–542, 2002.

[21] A. P. Rocha, H. Choupina, J. M. Fernandes, M. J. Rosas, R. Vaz, and J. P. S. Cunha, "Parkinson's Disease Assessment Based on Gait Analysis Using an Innovative RGB-D Camera System," in *36th Annual International IEEE EMBS Conference*. IEEE, 2014.

[22] InvenSense, "ITG-3200 product specification revision 1.7," 2011, accessed on 18/12/2013. [Online]. Available: http://www.invensense.com/mems/gyro/documents/PS-ITG-3200A.pdf

[23] Kionix, "± 2g / 4g / 8g tri-axis digital accelerometer specifications," 2011, accessed on 18/12/2013. [Online]. Available: http://www.kionix.com/sites/default/files/KXTF9-1026%20Specifications%20Rev%206.pdf

[24] Honeywell, "3-axis digital compass IC HMC5883L," 2013, accessed on 19/12/2013. [Online]. Available: http://107.23.57.149/sitecore/shell/Controls/Rich%20Text%20Editor/~/media/Images/Plymouth%20Website%20PDFs/Magnetic%20Sensors/Data%20Sheets/HMC5883L_3-Axis_Digital_Compass_IC.ashx

[25] D. Ribeiro, "Motion quantification Project."

[26] Oracle, "The History of Java Technology," accessed on 20/6/2014. [Online]. Available: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html

[27] Developer.android.com, "Developer Tools | Android Developers," 2014, accessed on 03/01/2014. [Online]. Available: https://developer.android.com/tools/index.html

[28] Bluetooth SIG, "A Look at the Basics of Bluetooth Technology," accessed on 8/11/2013. [Online]. Available: http://www.bluetooth.com/Pages/Basics.aspx

[29] Android, "Bluetooth API Guide," accessed on 10/11/2013. [Online]. Available: https://developer.android.com/guide/topics/connectivity/bluetooth.html

[30] Android, "Camera API Guide," accessed on 16/1/2014. [Online]. Available: https://developer.android.com/guide/topics/media/camera.html